Reinforcement-learning-based Cross Layer Design in Mobile Ad-hoc Networks

Wang Ke

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2015

Declaration

I hereby declare that the thesis is my original work

and has been written by me in its entirety.

I have duly acknowledged all the sources of information

which have been used in the thesis.

This thesis has also not been submitted for any degree

in any university previously.

Wang Ke

14th Aug 2015

To my wife, my parents and my son.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Prof. Lawrence Wong Wai-Choong and Dr. Chai Teck Yoong, for their continuous guidance, support and encouragement during my PhD candidature. Without their insights, knowledge, patience and enthusiasm this thesis's completion would not be possible.

I would also like to thank my thesis advisory committee members, A/Prof. Mohan Gurusamy and Asst Prof. Mandar Anil Chitre for their valuable suggestions during my PhD qualification examination.

I would like to acknowledge the financial, academic and technical support of National University of Singapore (NUS), and NUS Graduate School for Integrative Sciences (NGS).

I also would like to thank the NUS-ZJU Sensor-Enhanced Social Media (SeSaMe) Centre where this research is carried out. It is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media Programme Office.

I also appreciate the lab officers, Mr. Song Xianlin and Ms. Guo Jie, for all the assistance in my simulation setup. I would like to thank all my lab mates in IDMI Ambient Intelligence Laboratory: Wang Lei, Chen Penghe, Wang Jin, Yu Jinqiang, Zhang Li, Sun Wen and Guo Qingfeng.

Last but not the least, I would like to express my deepest thanks to my wife, my parents and my newborn son Layton. They are everything to me.

Contents

Summary

Mobile Ad-hoc networks (MANETs) are drawing increasing research interest because of the revolutionary development of mobile devices and wireless communication technology in recent years. One important feature of MANETs is that Nodes can move freely without any requirement of infrastructure. Therefore communications are purely based on peer-to-peer packet forwarding, which makes it an ideal option for fast network deployment such as in military or disaster areas. While mobility and lack of central controller are the most important features of MANETs, they are also the greatest challenges: constant changes in network topology and local information sharing make end-to-end Quality of Service (QoS) optimization difficult. In this dissertation, we propose a reinforcement-learning-based solution to address the cross-layer optimization on QoS performances in MANETs.

Our major work consists of three parts: Q-learning-based routing, power-controlled routing and rate-aware power-controlled routing. In the first part, we propose a self-learning routing protocol based on a Q-learning method that combines multiple QoS metrics as its reward, with the help of a congestion level indicator for the purpose of parameter tuning. As a result, nodes are able to dynamically change their routing strategies based on surrounding environment and previous experiences. In order to cope with the routing loop problem, we introduce

a probing packet mechanism to discover and resolve routing loops periodically.

In the second step, we add a power control scheme into the Q-learning method. To efficiently deal with the large action space formed by combining both routing and power control, a CSMA/CA delay model is adopted to accelerate the learning process by extracting a small number of variables from the environment, which are further processed to simulate all the routing and power-control actions. In order to find the optimal power level, a pricing mechanism based on interference level is also embedded into the system.

Finally, we consider one more factor: rate adaptation. Although it is possible to directly extend the power-controlled routing scheme by adding the rate-related parameters, a global-utility-based rate-aware power-controlled routing scheme based on the diffusion model is implemented to improve the performances. We also introduce a multi-agent coordination mechanism based on the Distributed Constraint OPtimization (DCOP) model. By comparing with various benchmarks in their respective categories, we show that our methods can achieve better performances with comparable energy consumption rate through simulation-based evaluations.

List of Figures

List of Tables

List of Symbols

Symbol	Meaning
α	Learning rate of the Q-learning method
β	Parameter of Boltzmann distribution
χ	Distance between two nodes
γ	Discount factor for future values
$\kappa_N^{(m)}$	Probability of m concurrent transmissions at one time plot among nodes N
λ	Packet arrival rate
μ	Average service rate
$\overleftarrow{Adj}$	Interfering neighbors
$\overline{\kappa}$	Adjusted probability of concurrent transmissions
$\overrightarrow{Adj}$	Interfered neighbors
$\Pr(\cdot)$	Probability of events
Ψ	Set of data flows
$\Re$	Real number
ρ	Collision probability of each transmission attempt
τ	Average transmission attempts per time slot
ε	Transmission range
$\widehat{\tau}$	Average transmission attempts of the neighbors
ξ	Rate parameter of the exponential distribution

ζ	Probability of being a mutual neighbor
A	Finite set of actions
Adj	Actual neighbors
b	State in Markov chain model
C	Congestion level
C_p	Cost function for power control
c_A	Coefficient of variance of the inter-arrival time
c_B	Coefficient of variance of the inter-service time
$CW_{\min}$	Minimal contention window size
d	Destination node
L	Delay
E	Set of edges
$E\{\cdot\}$	Expected value
F	Discount function which maps the value into the range of $(0, 1]$
G	Routing graph
$i,\ j,\ k,\ l,$	Indexes
I_p	Interference price
K	Mac-layer transmission queue length
L_p	Packet size
n_l	Pass loss exponential
o	Source node
p	Power level
P_b	Probability of channel busy
P_c	Probability of collision
P_s	Probability that a transmission attempt is successful
$p_{\max}$	Maximal power level

$p_{\min}$	Minimal power level
$P_a(s, s')$	State transmission probability
p_{ET}	Energy detection threshold
p_I	Interference power level
p_N	Noise power level
$R_a(s, s')$	Immediate reward
r_G	Energy gain
$U(\cdot)$	Data rate
$U_{\min}$	Minimal data rate
$U_{\max}$	Maximal data rate
r	Packet generating rate
$R(\cdot)$	Routing strategy
S	Finite set of states
t	Iteration index
T_{SINR}	Threshold of SINR value required
V	Set of nodes
w	Weight for reward function
X	Backoff delay without the counter pause time

List of Abbreviations

Abbreviations	Full Name
AARF	Adaptive ARF
ACK	Acknowledge
AI	Artificial Intelligence
AODV	Ad hoc On-demand Distance Vector
AODV-HPDF	AODV routing protocol with High Packet Delivery Fraction
AOMDV	Ad hoc On-demand Multipath Distance Vector
ARF	Automatic Rate Fallback
BER	Bit Error Rate
CARA	Collision-Aware Rate Adaptation
CBR	Constant Bit Rate
COMPOW	COMmon POWer
CPC-AODV	Cross-layer Power Control Ad hoc On-demand Distance Vector
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
DBA	Distributed Breakout Algorithm
DCF	Distributed Coordination Function
DCOP	Distributed Constraint OPtimization
DIFS	DCF Interframe Space

DISPOW	DIStributed POWer management
DSA	Distributed Stochastic Algorithm
DSDV	Destination-Sequenced Distance-Vector Routing
DSR	Dynamic Source Routing
DTPC	Dynamic Transmission Power Control
EIFS	Extended InterFrame Space
FAR	Full Auto Rate
GA	Generic Algorithm
GRACE	Gradient Cost Establishment
IEEE	Institute of Electrical and Electronics Engineers
LET	Link Expiration Time
LMA	Local Mean Algorithm
LMN	Local Mean of Neighbors algorithm
MAC	Media Access Control
MAENT	Mobile Ad-hoc Network
MDP	Markov Decision Process
MRPCR	Multi-agent Rate-aware Power-Controlled Routing
NS	Network Simulator
OSI	The Open Systems Interconnection
PARO	Power-Aware Routing Optimization
PCM	Power Control MAC
PCON	Power sensitive power CONtrol
PCR	Power Control Routing
QLAODV	Q-Learning AODV
QLMAODV	Q-Learning MAODV
QLPCR	Q-Learning-based Power-Controlled Routing

QLR	Q-Learning-based Routing
QLRPCR	Q-Learning-based Rate-aware Power-Controlled Routing
RBAR	Receiver Based Auto Rate
RRAA	Rate Adaptation Algorithm
RREP	Route Reply
RREQ	Route Request
RTS	Request To Send
SIFS	Short InterFrame Space
SINR	Signal-to-Interference-plus-Noise Ratio
TPCRA	Transmission Power-Controlled Rate-Aware
WMN	Wireless Mesh Network

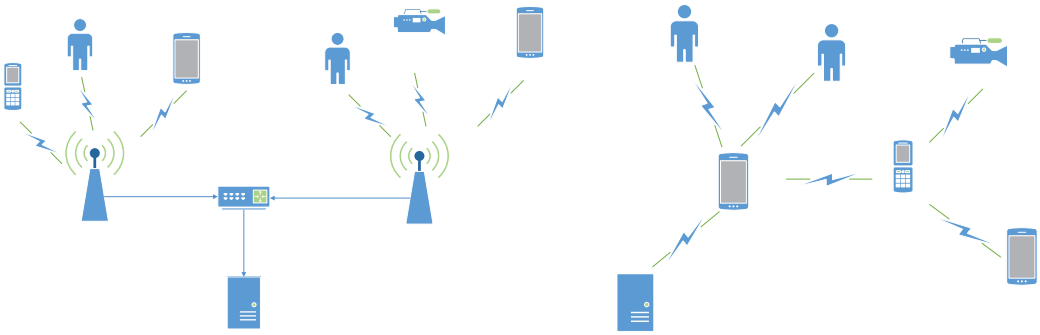
Chapter 1

Introduction

1.1 Background

1.1.1 Mobile Ad-hoc Network

The recent development in wireless-medium-based communications has dramatically changed people's life. Wireless networks can be generally divided into two main groups: infrastructure-based and infrastructure-less wireless networks (Fig. 1.1). The most famous member in the first group is the cellular network. Powered by multiple cellular radio towers and wired infrastructure, the cellular network is capable of providing large radio coverage and high throughput to its end users, with the requirement that communications are always between the end users and infrastructure. Since the end users do not rely on each other for packets forwarding, requests from them are forwarded to destinations through the backbone infrastructure. As a result, the end users can enjoy the freedom of the wireless communication while avoiding the shortcoming of unreliable wireless communication channels for long distances.



(a) Infrastructure-based wireless networks (b) Infrastructure-less wireless networks

Fig. 1.1: Infrastructure-based vs. Infrastructure-less wireless networks

Infrastructure-less networks, such as wireless ad hoc networks have advantages on flexibility, scalability, easy deployment and relatively low cost. They do not rely on the infrastructure, and therefore are suitable for emergency situations such as disaster recovery or military usage. Wireless Mesh Networks (WMNs) combine both features of infrastructure-based and infrastructure-less networks by meshing client access gateways through ad hoc connection. Mobile Ad hoc Networks (MANETs) are a special kind of wireless ad hoc networks (Fig. 1.2) with an extra restriction that its end users consist of mobile devices only. In fact, restrictions in MANETs are quite limited: they can choose to be cooperative, selfish, invisible or even destructive (although may not be on purpose). How to organize all the nodes in MANETs to accomplish a certain task is still an open challenge because many well-addressed problems in wired networks, such as routing, rate adaptation and power control cannot be applied directly to MANETs.

One major challenge comes from mobility. Frequent changes in topology require an efficient information sharing and coordination system. However, limited communication range and lack of a central controller make the system design challenging: communication in MANETs involves intermediate nodes exchanging

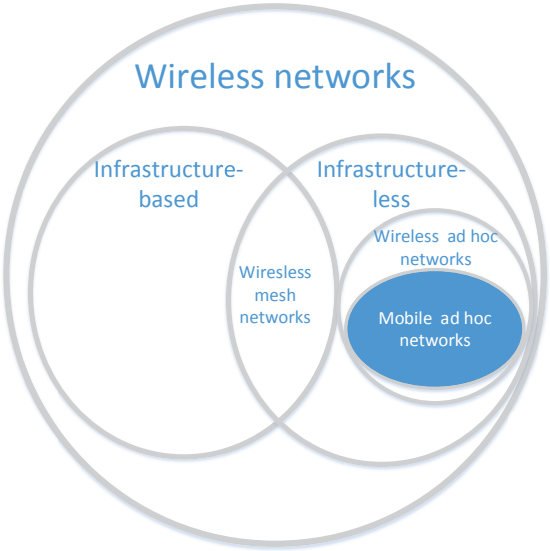


Fig. 1.2: Types of wireless networks

local information. As a result, excessive information sharing introduces undesirable overhead, which may further impair the whole network performance. On the other hand, limited information-sharing bandwidth prevents the traditional graph-theory-based algorithms from working on MANETs. Therefore, compromising the trade-off between information sharing and overhead is the key to good design for MANETs.

Limited lifetime of mobile devices is another challenge faced by MANETs. Efficiently forwarding packets becomes crucial for energy saving purpose. One important fact is that, it is not necessarily true that smaller transmission power results in smaller power consumption. In fact, higher transmission power may shorten the overall transmission time if a higher rate can be adopted, which may eventually reduce the power consumption.

In wireless networks, Quality of Service (QoS) measures the overall performance observed by an end user. Unfortunately, it is usually the bottleneck of the development of MANETs because wireless communication channels are less

reliable than wired ones due to multi-path propagation and channel fading. As a result, when the hop counts of a packet increases, the probability of transmission failure increases significantly. The resulting poor QoS performances limit the applicability of MANETs: high packet loss cannot guarantee critical information delivery; high latency rules out real-time communications; low throughput makes multimedia applications almost impossible. This is also the reason why MANETs become an inferior choice if an infrastructure-based solution is possible.

However, MANET has a large room for improvement thanks to its flexibility: nodes in MANETs are capable of changing its power level, rate adaptation method and routing strategy to accomplish packet-forwarding tasks on demand. Therefore, with a carefully designed coordination system, the overall QoS performances can be greatly boosted.

Before we explain this, let us give an brief introduction of power control, rate adaptation and routing respectively.

1.1.2 Power Control

In a wireless network, nodes are connected through wireless data connections. A wireless signal is generated by a digital modulation method, and then transmitted by an antenna, which is capable of converting electric power into radio waves. In order to successfully receive the signal, the signal strength at the receiver side has to exceed the energy detection threshold p_{ET}. The required transmission power p and signal gain r_G for a successful transmission can be expressed as

$$p \cdot r_G \geq p_{ET} \tag{1.1}$$

Note that (1.1) is only a necessary condition for a successful transmission. In order to correctly demodulate the received signal, a sufficient Signal-to-Interference-plus-Noise Ratio (SINR) for a given modulation method is also required, which is denoted as

$$\frac{p \cdot r_G}{p_N + p_I} \geq T^U_{SINR} \tag{1.2}$$

where p_I and p_N are the interference and noise power strength, respectively. T^U_{SINR} is the threshold of SINR value required for a data rate U. Note that fulfilling (1.1) and (1.2) does not guarantee a successful transmission attempt in practice. However, in such a case, either the Bit Error Rate (BER) is negligible or the error bits can be recovered by using some error correction techniques. Therefore, for simplicity, we assume perfect reception when the energy detection threshold and the SINR requirements are satisfied.

The solution of the power control problem is to find the best power level to achieve good performances within the network. The challenge comes from the fact that higher U requires higher SINR, which in turn requires higher transmission power. However, higher transmission power increases p_I at the same time. Another important side effect of the power control scheme is the change of network topology, which has great impacts on routing decisions. Therefore, it is preferred that the power control scheme can take surrounding traffic conditions into consideration. For example, a larger transmission range has a great advantage in a low-traffic multi-hop network because it can shorten the path length and reduce the end-to-end latency. However, in high-traffic conditions, the cost of the packet forwarding becomes significant because of the increased contention among neighbors. In such a scenario, a smaller transmission power is preferred.

1.1.3 Rate Adaptation

The main purpose of rate adaptation is to achieve higher throughput, with major challenges of accurately assessing the channel condition and differentiating packet loss due to contention from that due to poor channel conditions [1]. In other words, rate adaptation is the determination of the optimal data transmission rate most appropriate for current wireless channel conditions [2]. Generally speaking, higher data rate leads to higher packet loss, it is therefore crucial to find the best trade-off between data rate and packet loss that maximizes the overall throughput.

1.1.4 Routing

Routing, as its name suggests, is to find the best route to a certain destination. Different routing protocols have different definitions of the "best": minimum hops, minimum power consumption, best link quality are all candidates of the routing metrics. Whichever is chosen, there will be shortcomings and trade-offs. The choice of routing protocol therefore depends on the scenarios, user preferences and applications. The most widely used method for the routing problem is the graph theory in which the whole network can be generalized as a graph $G = (V, E)$ consisting of vertex V and edges E. In MANETs, the vertexes become mobile nodes and the routing problem can therefore be converted into a shortest path problem: find a path between two vertexes that minimizes the sum of the weights of its constituent edges.

Bellman-Ford [3, 4] is an algorithm that finds the short path from one source to the rest of the nodes based on the principle of relaxation. At first, the distance of each node to the source is marked as infinity. At each iteration, the distances are relaxed when a shorter distance is found. The general idea can be represented as a dynamic programming approach shown in Fig. 1.3: in each iteration, the

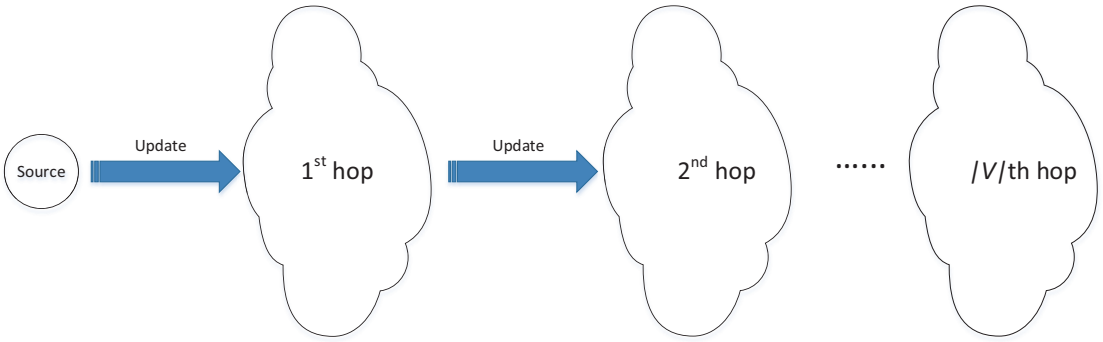


Fig. 1.3: Bellman-Ford shortest path algorithm. In each iteration, source will move forward for one more hop, thus at the ith iteration, the shortest distances to all the nodes by using up to i hops are found

relaxation process moves one hop forward until there is convergence. Note that one node can be updated multiple times. Since the maximal number of hop is $|V|$, It is guarantee that after $|V|$ iterations the convergence is reached. For each iteration we have to go through all the edges, which results in the overall complexity of $O(EV)$. When the algorithm finishes, the shortest path from the source to all the nodes in V is obtained. If hop count is chosen as the routing metric (in this case, all the edge cost equals to one), the algorithm can terminate once the destination is reached for the first time.

Dijkstra's algorithm [5] solves the shortest path problem by maintaining a set of nodes S whose shortest path is already found. Then the algorithm repeatedly adds the vertex in $V - S$, whose path estimate is the shortest, into set S. The process terminates when $S = V$. For example, in Fig. 1.4, at a certain iteration, there are three edges connecting S and $V - S$, Dijkstra's algorithm chooses the node in $V - S$ with the smallest distance from the source, in this case, node D, and adds it into S. Unlike the Bellman-Ford algorithm, if any node is added into set S, its distance will not be updated any more. Therefore, the algorithm can terminate once the destination is added into S.

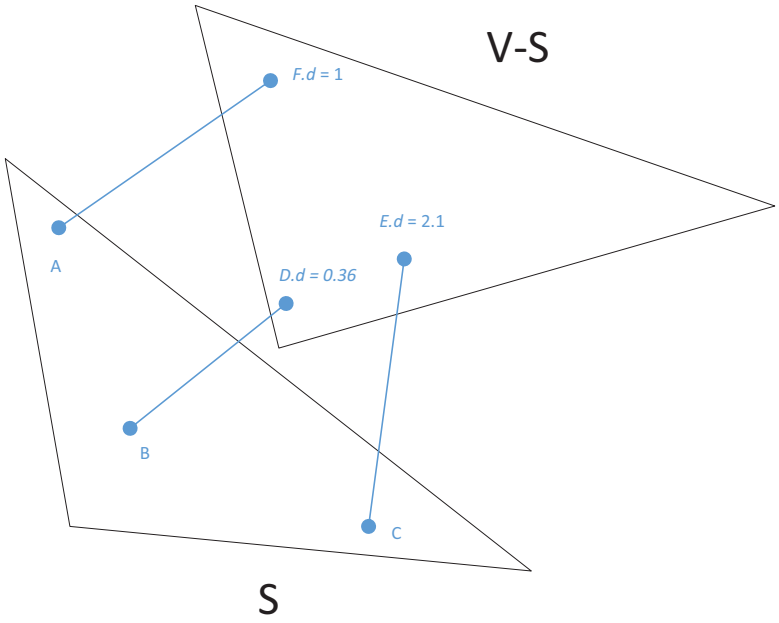


Fig. 1.4: Dijkstra shortest path algorithm

Introducing heuristics can further extend the Dijkstras algorithm. A* search algorithm proposed in [6] is an efficient path-finding algorithm by estimating future cost heuristically. The A* algorithm calculates a heuristic cost to reach the destination in addition to the cost incurred from the source. For example, we can use the Euclidean distance as the heuristic function in A* to find shortest walking distance from a source to a destination. The cost of an intermediate node consists of two parts, one is the actual cost measured by the shortest path from the source to the current node, the other is the estimated cost from the current node to the destination, which, in this case, is the length of the line connecting the current node and the destination. Note that in the example, the estimated future cost is always smaller than the actual future cost, therefore, the heuristic function is admissible. It is guarantee that when the heuristic function is admissible, the A* algorithm can always find the optimal path.

Among the three algorithms, A* search has the lowest running time, in spite of

the difficulties in finding an admissible heuristic function, especially when location information is expensive or unreliable. The Dijkstra's algorithm is running faster than the Bellman-Ford algorithm because of its greedy approach in including the finished nodes. However, it requires that all the cost of edges must be positive. Even worse, it must rely on the global topology information to make decisions. On the contrary, the Bellman-Ford algorithm, although having the longest running time, can be distributed so that each node just updates its shortest distance from the source whenever possible. More importantly, even when the optimal solution has not been found yet, the algorithm can still provide a suboptimal path leading to the destination.

The largest disadvantage of a graph-based approach is that it can only work in a static environment with a fixed topology. In MANETs, during the execution of a graph-based algorithm, the graph itself may already change due to link failure or node movement. Therefore, it is crucial for a routing protocol in MANETs to provide a satisfactory any-time solution, which also has the capability of evolving over time.

1.2 Motivation

Interactions between power control, rate adaptation and routing can be generalized as in Fig. 1.5. Power level directly determines not only transmission range but also the corresponding maximum supported data rate. Based on current SINR level, the rate adaptation scheme chooses the best data rate accordingly. The topology formed by the power control and rate adaptation scheme is then pass to the routing protocol. At the same time, since the routing protocol chooses only a subset of nodes as intermediate nodes, it is also important for the power

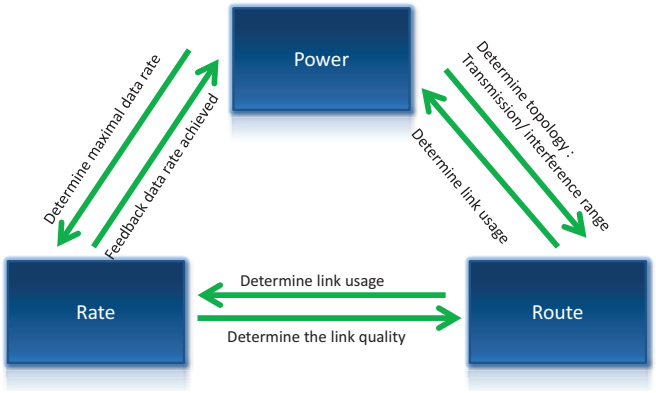


Fig. 1.5: Interaction between power, rate and route

control and rate adaptation scheme to be aware of the selected edges so that the optimization can be carried out.

A power-controlled rate-aware routing protocol design can be straight forward if following the layered structure in Fig. 1.5. For example, if we want to minimize the overall energy consumption, we can use the minimal power level to maintain the network connectivity. Then the algorithm chooses lowest data rate to ensure maximum transmission range, and then forwards packets along the path with minimum power consumption. If the utility function is to maximize the overall throughput, we can start with rate adaptation by using the highest data rate possible. The power level should be large enough to support the links to its highest data rate, if not possible, then maximal power level is adopted. Finally, the protocol can make use of a maximal flow algorithm to obtain the optimal path, or simply chooses the next-hop data rate as one of the routing metrics to find the shortest path. For both of the examples, the general ideas behind are the same: for a given utility function, we can try to optimize all of three factors separately and combine them together to form a complete solution. However, such an approach does not fully utilize the flexibility and exploit the potential of MANETs.

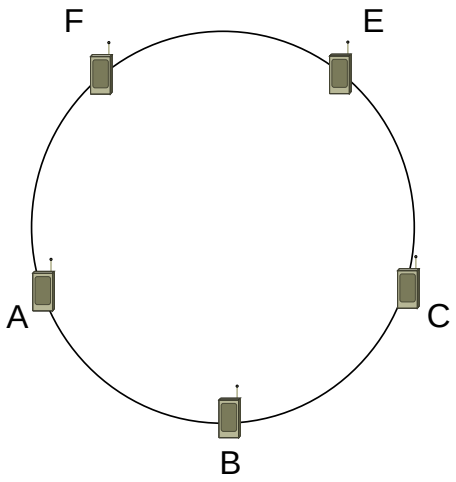


Fig. 1.6: Design motivation. Five nodes are connected in a ring, node A is sending packets to node C

Using a simple example, it is easier to illustrate the fact that the cross-layer design provides optimization options that no single-layer optimization can achieve. Fig. 3.9 shows a MANET with five nodes connected in a ring with traffic flowing from node A to node C. If only routing is considered, it is generally a good choice for node A to take the shortest path $A-B-C$ because fewer intermediate nodes are involved.

Now let us assume that node B is far away from both node A and node C. Fig. 1.7 indicates the bandwidth for each link. In this case, going through node F and node E may eventually result in higher throughput. Therefore, with the extra information from another layer, the decision can be totally different.

The cross-layer design is not limited to the extra information, new actions can also be introduced when multiple layers are optimized together. For example, suppose now each node can freely adjust its power level and node C is within node A's maximum coverage, i.e., node C can be reached by node A by using the maximum transmission power. Under such conditions, The best choice for node

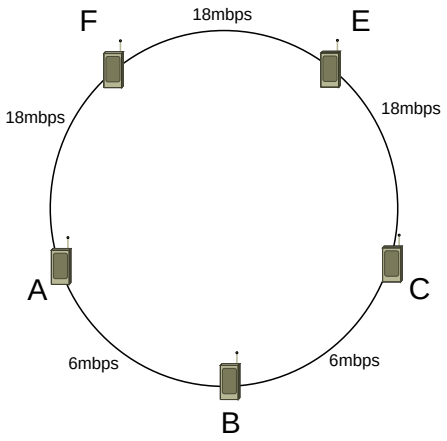


Fig. 1.7: Routing with rate information. The maximum rates are labeled in the figure, it is therefore a better choice to go through a longer path. However, if nodes can freely adjust their power levels, it might be a good choice for node A to directly reach node C by increasing its power level.

A would be to increase the transmission power to reach node C (as shown by the dotted line in Fig. 1.7) without involving any intermediate node. Such an action cannot be made if the power control or the routing strategy operates separately.

Fig. 1.7 is a simplified graph-theory-based routing model which ignores the transmission interference caused to neighbors. In reality, an Omni-directional antenna will affect all nodes within its transmission range by introducing extra interference. As a result, the supported maximum data rate drops due to the reduced SINR and more intensive contention. Taking Fig. 1.7 as an example, the power level required for node A to reach node C also makes node E under its transmission range. Therefore, the link $E - F$ and $E - C$ now suffer throughput loss due to the increased interference. If there is heavy traffic on these links, the resulting performance loss can be significant. The decision on whether to use a larger transmission power therefore depends on the trade-off between the performance improvement of itself and the interference level caused to its neighbors.

This example explains one important fact: with more useful information and

options, nodes have more chances to make a better choice, although it is tough. The Open Systems Interconnection Model (OSI Model) [7] divides the communication system into seven abstract layers so that each layer can be developed independently. It saves a lot of trouble for the protocol designers since they can focus on one layer at a time. By breaking the layered model as in the previous example, we must face the challenge of how to use the extra information to obtain better performances. Unfortunately, it is not an easy task because multiple layers have complicated impact on the final outcomes.

1.3 Problem Definition

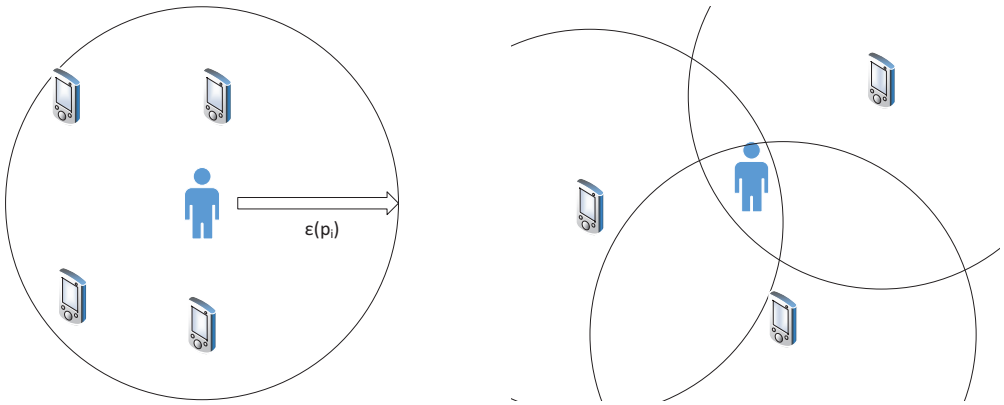
Before getting into the solution, let us define the problem first. A network adopting the IEEE 802.11 Distributed Coordination Function (DCF) Media Access Control (MAC) protocol, consists of a set of nodes V and a set of data flows Ψ. Each flow $\psi_i \in \Psi$ can be represented as a tuple (o_i, d_i, r_i), where o_i and d_i are the source and destination node and r_i is the packet-generating rate. Let p_i be the transmission power level for node i such that $\forall p_i : p_{\min} \leq p_i \leq p_{\max}$, where $p_{\min}$ and $p_{\max}$ are the minimal and maximal power level for a mobile node. Typically, p_i is a discrete variable for most mobile devices. Let $\varepsilon(p_i)$ be the transmission range for a given power level p_i and $\chi_{i,j}$ be the distance between nodes i and j. Then we can define the interfered neighbors of node i as $\overrightarrow{Adj}_i$, which is the set of nodes that are within the sender node i's transmission range. Symmetrically, the interfering neighbors set is denoted as $\overleftarrow{Adj}_i$, which represents the set of nodes whose transmission range can reach node i (Fig. 1.8). The reason of maintaining the two sets of neighbors is the asymmetry introduced by the power control scheme: since all nodes can change their transmission power levels, it is quite possible that

$\varepsilon(p_i) \neq \varepsilon(p_j)$ for any two given nodes. Therefore, it is necessary to differentiate $\overrightarrow{Adj}_i$ from $\overleftarrow{Adj}_i$ since they may be different as well. Lastly, we denote the actual neighbor set as $Adj_i = \overrightarrow{Adj}_i \cap \overleftarrow{Adj}_i$. Transmissions from node i can only succeed if the receiver is in Adj_i because of the handshake procedure introduced in the IEEE 802.11 MAC protocol. Note that both $\overrightarrow{Adj}_i$ and Adj_i are functions of transmission power p_i. The routing strategy for node i to destination node d can be denoted by $R_i(d)$. The consecutive strategy therefore is defined as $R_i^{(t)}(d)$ where $R_i^{(t)}(d) = Rt_{R_i^{(t-1)}(j)}(d)$ and $R_i^{(1)}(d) = R_i(d)$. $U_{i,j}(p_i)$ represents the expected data rate from node i to node j by using power level p_i. $f(\psi_i)$ is the utility function for traffic flow ψ_i. The optimization problem can be represented as:

$$\begin{aligned} \max\ f(\Psi) &= \sum_{i \in N} f(\psi_i) \\ \textit{subject to } p_{\min} \leq p_i \leq p_{\max}, i &= 1,2,3,4...v \in V \\ R_i(d) &\in Adj_i(p_i), i = 1,2,3,4...v \in V \\ U_{\min} \leq U_{i,j}(p_i) &\leq U_{\max}, j \in Adj_i(p_i) \\ R_i^{(t)}(d) &= d,\ \textit{where } t < \infty \end{aligned}$$

Modeling power level, rate adaptation and routing mathematically at the same time is difficult because of the non-linear constraints of the actual neighbor set and the optimal data rate. Common approaches such as the graph theory method require a well-defined topology because routing does not make much sense when edges do not exist. On the other hand, constructing an optimal topology also requires the information of routing because only the links participating in for packet forwarding need be optimized.

Artificial Intelligence (AI) techniques such as reinforcement learning method



(a) Interfered neighbors (b) Interfering Neighbors

Fig. 1.8: Interfered neighbors vs. Interfering Neighbors

could be a possible solution since it allows agents to explore and learn from the environment without a system model. The Q-learning method is a model-free reinforcement learning technique for optimal policy search in any given Markov decision process (MDP) [8]. By starting with some initial settings, each node can adjust its power, rate and routing strategy based on its own observations.

1.4 Literature Review

1.4.1 Power Control

Power control is critical because of its impact on transmission range and battery life [9]. Extensive studies have been carried out in the area of power control in cellular systems [10–13], which mainly focus on single-hop transmissions. In MANETs, it is common that packet traverses more than one hops to reach the destination, therefore, the power control scheme has to pay more attention to SINR and space reuse ratio. Controlling power level in a multi-hop network was first introduced by Kleinrock et al [14] with the constraint that all nodes must

use the same transmission power level. Hou et al [15] remove the restriction by introducing an analytical model for throughput and forward progress using variable transmission power levels. ElBatt et al [16] combine power control and scheduling by packing the maximum number of transmissions into the transmission packets which determines the set of powers that could be used by the scheduled users. It requires a controller responsible for executing the scheduling algorithms. Long et al [17] develop a non-cooperative reinforcement-learning-method-based power control algorithm with the main objective of minimizing the transmitting power. It provides a power control mechanism based on a stochastic fictitious play model with repeated games. Genetic Algorithm (GA) can also provide a solution to the power control problem. Wu et al [18] propose a permutation encoded GA by formulating the minimal power broadcast problem to a graph-based constrained optimization problem.

Topology control is another kind of power control scheme. The topology of a multi-hop wireless network is the set of communication links between node pairs used explicitly or implicitly by a routing mechanism [19]. Unlike the power control scheme in the cellular network, topology control in MANETs mainly focuses on the connectivity rather than the quality of each link. In other words, it tries to maintain the number of neighbors within a reasonable range. The Hybrid and the AEWMA scheme described in [20] changes the transmission power level in response to the neighbors' piggybacked power query message. The Power Control MAC (PCM) protocol in [21] and Power-Aware Routing Optimization (PARO) in [22] use the maximal power level for the Request To Send (RTS) and Clear To Send (CTS) frames and a much lower power level for data frames. The Local Mean Algorithm (LMA) illustrated in [23] tries to maintain the number of neighbors between $NodeMinThresh$ and $NodeMaxThresh$. If the number of current

neighbors is smaller than $NodeMinThresh$, the power level will be increased. Likewise, power level will be reduced when the current number of neighbors is larger than $NodeMaxThresh$. The Local Mean of Neighbors algorithm (LMN) in [23] does not have a fixed threshold. Instead, each node tries to keep its number of neighbors as the same as the mean value of its neighbors' number of neighbors. The Common Power (COMPOW) algorithm in [24], as its name suggests, tries to find a common power level for all nodes such that the entire network remains connected. The DIStributed POWer management (DISPOW) described in [25] sends out power-change requests based current connectivity, interference and energy level, then adjusts power level according to the current status and the power requests received. All these approaches are able to construct a reasonable topology with most nodes connected by a relatively small power level. However, the optimal performance cannot be obtained because the impact of power control on routing and rate adaptation is overlooked.

1.4.2 Rate Adaptation

The rate adaptation schemes can be divided into two groups based on their support of loss differentiation. The schemes without loss differentiation adjust their data rate according to frame loss or signal strength. For example, Kamerman et al [26] propose a scheme which chooses a higher transmission rate after a number of successful transmissions at a given rate and switches back to a lower rate after several consecutive failures. The Adaptive ARF (AARF) scheme proposed in [27] follows a similar idea but dynamically chooses the threshold for rate switching. The Receiver Based Auto Rate (RBAR) scheme proposed in [28] is the first rate adaptation scheme adjusting the transmission rate based on the channel information obtained through MAC level control message. The Opportunistic

Auto Rate (OAR) scheme in [1] is similar to RBAR with an extra feature of the opportunistic transmission of data frames. The Full Auto Rate (FAR) scheme in [29] increases the data rate of the RTS / CTS to achieve better performance.

The other group is the rate adaptation schemes with a loss differentiating mechanism, which is used to diagnose the cause of a packet loss. Packet loss can be caused due to channel degradation or packet collision. Only in the first case, rate should be decreased in order to reduce the packet loss rate. Collision-Aware Rate Adaptation (CARA) in [30] uses the RTS packets to probe the channel condition when encountering packet loss. Since RTS / CTS packets are always sent at the basic rate, it is not quite possible that the packet loss is due to collision if the RTS / CTS packets are lost as well. The Robust Rate Adaptation Algorithm (RRAA) scheme in [31] combines both packet loss count and MAC-layer information to estimate short-term loss ratio and opportunistically guides its rate change decisions.

Most of the rate adaptation schemes are based on the MAC layer. In fact, some power control and rate adaptation schemes, such as [32–34], also mainly focus on the MAC level one-hop single-flow scenarios.

1.4.3 Routing

Routing is a path selection process based on a predefined topology. One or more metrics need to be chosen as path weight for routing process. Most of the conventional routing protocols are designed either to minimize the data traffic in the network or to minimize the average hops for delivering a packet [35]. For example, Ad hoc On-demand Distance Vector (AODV) [36], Dynamic Source Routing (DSR) [37] and Destination-Sequenced Distance-Vector Routing (DSDV) [38] are all shortest-path-algorithm-based routing protocols that are well known

for their simplicity and fair performances. However, since they only consider the hop count when selecting a route, the route selected may not be the optimal one in terms of QoS parameters such as delay and throughput. Besides, the selected route may fail since the hop count cannot guarantee any stability and reliability.

Many prior works have been done to improve the original AODV algorithm. The AODV routing protocol with High Packet Delivery Fraction (AODV-HPDF) protocol in [39] utilizes local repair at the upstream intermediate node by introducing "the salvage process", during which the intermediate node tries to send out packets buffered to the destination as a source. At a given scenario, its packet delivery rate can be slightly increased if these originally dropped packets can be successfully retransmitted. However, in a highly dynamic environment where link failures are frequent, such a salvage process may introduce significant overhead. Adding multi-path routing features into AODV is another possible solution. Marina and Das [40] propose the Ad hoc On-demand Multipath Distance Vector (AOMDV) routing protocol which tries to buffer potential routes so that when a link breaks, backup routes can be used directly without sending extra routing control messages. However, such backup routes still suffer from node mobility [41].

Some researchers also try to use reinforcement learning methods to solve the QoS optimization problem in MANET because AODV's packet forwarding policy is purely based on the hop count. Among all these available learning methods, Q-learning is preferred because it does not require a model of the environment. Chang and Kaelbling [42] propose a scheme using the reinforcement learning methods to control both packet routing decisions and node mobility to improve the connectivity of the network. However, the assumption that node mobility can be controlled by its routing protocol is usually not valid for MANETs [43].

Q-Learning AODV (QLAODV) proposed in [43] is another MANET routing

protocol that considers link stability and bandwidth efficiency. It uses distributed Q-learning to infer network status information and takes link stability and bandwidth efficiency into consideration while selecting a route. For each node, its bandwidth factor and mobility factor are calculated and broadcast to its neighbors through the AODV Hello messages. Both factors are used to update the discount factor in Q-learning. Therefore, for a given node, all its neighbors will receive the same discount factor, which is not fair because obviously some neighbors are closer or have better link quality than others. Besides, the term "maximum bandwidth" used to calculate the bandwidth factor cannot be easily obtained since it depends on the topology of MANETs. Q-Learning MAODV (QLMAODV) follows the same idea, which combines mobility, bandwidth and residual power together to calculate the discount factor. However, the mobility factor requires the exact location and moving pattern for each node which is not easy to obtain.

In summary, the Q-learning-based routing schemes proposed in [39, 42–46] can easily combine different routing metrics by treating the entire network system as a black box. However, using multiple metrics inevitably involves weight tuning. Moreover, the final result may be trapped in a local optimal point due to the lack of peer-to-peer coordination and the system model.

1.4.4 Power-controlled Routing

There are two types of protocols that take power control and routing into account at the same time. The first category is the energy-aware routing protocols, which use remaining energy level or power consumption rate as the path weight to reduce energy consumption and prolong network life time, such as [47–49]. They do not control transmission power levels directly.

The second category, however, explicitly changes the transmission power level

for performance improvement. One major challenge for the power-controlled routing protocol design is to cope with the complicated interaction between power level and routing strategy. A feasible solution is to combine existing power control protocols and routing protocols to bypass this challenge as in [50–52]. However, overlooking the interaction between power control and routing has performance penalty as described earlier. Therefore, many researchers propose delegated power-control routing protocols. Power Control Routing (PCR) proposed in [53] explicitly tries each power level to find the best combination of power control and routing that minimizes the overall path cost. Cross-layer Power Control Ad hoc On-demand Distance Vector (CPCAODV) in [54] builds different routing entries for different power levels on demand, and selects the minimum power level for data delivery. TopoLogy-control-based QoS Routing (TLQR) in [55] uses hop count based on residual bandwidth for different power levels. The residual hop count is estimated by each node's two-hop neighbors' used bandwidth, distance, and power level. Since all these protocols require each power level be explicitly tested, they may not be affordable, especially in a dynamic network with a large number of nodes.

1.4.5 Power-controlled Rate-aware Routing

Power-controlled Rate-aware routing is a relatively new area and therefore less work has been carried out. Kwon and Shroff [56] propose such a scheme allocating links to incoming flows. The algorithm solves an optimization problem to minimize the average energy consumption while maintaining SINR constraints. However, the algorithm makes an unrealistic assumption about total control over node scheduling on a static topology without contention. The most relevant work is the Transmission Power-Controlled Rate-Aware (TPCRA) routing protocol in

[57], which develops a MAC-level module to support the calculation of the transmission power and data rate of a neighborhood in a scalable way. At first, each node estimates the ideal transmission power and modulation to all its neighbors through message broadcasts. Then, an objective function considering data rate, path length and power level is evaluated to construct the routing paths. One major drawback of TPCRA is that the extra interference and contention introduced by higher power level and routing path is not considered. Moreover, when constructing the topology, routing is not jointly optimized with rate adaption.

1.5 Thesis Contribution

The Q-learning-based method bypasses the complicated system model by introducing a self-evolving process consisting of acting, observing, learning and updating. However, Q-learning has its own defects when applied to MANETs, such as static environment restriction, routing loop problem, selfish behaviors, slow convergence rate and lack of coordination, etc.. In this thesis, we propose a Q-learning-based framework that jointly optimizes power control, rate adaptation and routing, meanwhile customized to overcome the problems in the traditional Q-learning method.

To be more specific, the main contributions of this thesis are listed below:

- Design a routing protocol which uses a distributed multi-metrics Q-learning algorithm to find the optimal path
- Implement an automatic parameter tuning mechanism to differentiate various traffic conditions based on congestion level and a learning parameter reset mechanism to deal with mobility
- Introduce a probing packet system for routing loop detection and recovery

- Design a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) delay model to overcome the large action space formed by combining power control and routing
- Measure the interference level in terms of weighted delay and use the interference level to derive an optimal transmission range.
- Implement a global-utility-based routing protocol based on the diffusion model
- Design a power-controlled rate-aware routing protocol using a global utility cost function
- Develop a multi-agent coordination scheme based on the Distributed Constraint OPtimization (DCOP) framework

1.6 Thesis Outline

We provide the solution of jointly optimizing power control, rate adaptation and routing in MANETs using the reinforcement-learning-based method in three steps. Since considering all three factors at the same time is challenging, we just add one factor at a time. Firstly, we illustrate how the Q-learning method can be applied to the routing problem. Then, a power control scheme, which deals with the trade-off between interference and transmission range, is embedded into the system. Finally, a rate adaptation scheme is added to form the complete solution. In the following content, Chapter 2 presents a Q-learning-based routing protocol that chooses the optimal path based on multiple metrics. Chapter 3 combines power control and routing protocol together, with the help of a CSMA/CA model,

so that the learning process is instantaneously completed for all neighbors. Chapter 4 adds the rate adaptation into the protocol. Inspired by the Braess's paradox, the global utility, which is obtained from diffusion model, is used as the instant rewards in the Q-learning framework. Besides, a coordination mechanism based on the DCOP framework is also developed to address the multi-agent learning problem. Finally, Chapter 5 summarizes the whole thesis and indicates possible extensions as the future work.

Chapter 2

Q-learning-based Routing Protocol

One distinguish feature of reinforcement learning is its emphasis on learning by the individual from direct interaction with its environment, without relying on exemplary supervision or complete models of the environment [58]. A typical application of reinforcement learning is in finding the exit of a maze without prior knowledge of the exact layout. A learning agent has to find a path to the exit in order to maximize its utility (assuming each step provides some reward along the path). In order to obtain desired performances, the reward mechanism needs to be carefully designed. For example, if the goal is to find the shortest path to the exit, then we can give each step negative reward whereas give the exit large positive reward. On the other hand, if the reward mechanism is poorly designed such that the maze contains a loop with positive reward, then it is quite likely that the learning agent will be trapped in the maze forever.

Routing is also an appropriate application of reinforcement learning because

finding a route to a given destination is conceptually the same as finding the exit in an unknown maze. In this chapter, a short introduction of reinforcement learning and Q-learning is given at first. Then we present the Q-Learning-based Routing (QLR) scheme by showing how the Q-learning method can be applied to the routing problem using multiple QoS metrics, followed by some customized optimizations techniques, such as a congestion-level-based parameter tuning scheme. QLR also sends out probing packets to detect and solve the routing-loop problem which is not addressed in most existing Q-learning-based routing protocols. Finally, we use simulations to evaluate our protocol under different scenarios.

2.1 Reinforcement Learning and Q-learning

A typical application of reinforcement learning is in a Markov Decision Process (MDP) [59], which provides a mathematical framework for modeling decision making in a stochastic environment. MDP can be considered as a 4-tuple: $(S, A, P_a(s, s'), R_a(s, s'))$, where S and A are the sets of states and actions, $P_a(s, s')$ is the transition probability that state $s \in S$ changes to state $s' \in S$ when action $a \in A$ is performed, with an immediate reward $R_a(s, s')$. Note that it is not necessary that $s \neq s'$. Fig. 2.1 is an example of a MDP model with s_0 and s_3 being the starting and ending state respectively. The numbers on the arrows indicate the transmission probabilities between different states. The whole process terminates when the ending state is reached. The goal of a learning agent in the MDP model is to find an optimal policy that maximizes the aggregated reward.

If both the transition probabilities $P_a(s, s')$ and rewards $R_a(s, s')$ are known, it can be solved by the Dynamic Programing (DP) technique [60]. However, if $P_a(s, s')$ or $R_a(s, s')$ is unknown, it becomes a reinforcement learning problem [61].

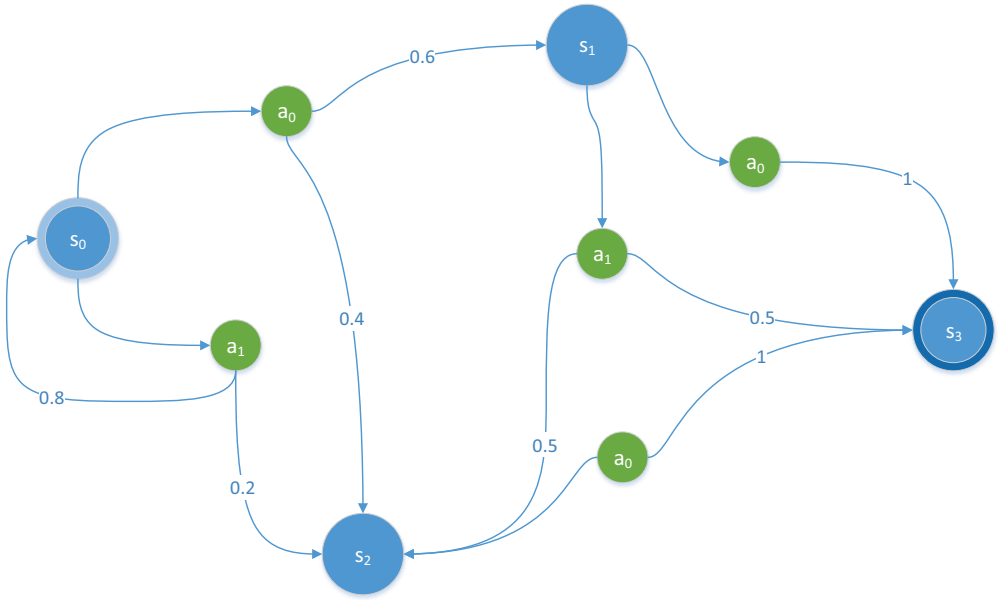


Fig. 2.1: Markov Decision Process. Node s_0 and s_3 are the starting and ending states respectively, a_0 and a_1 are the actions available at each state, the numbers on the arrows indicate the transition probability $P_a(s, s')$

In reinforcement learning, the Q value is introduced to indicate the optimality of an action in a given state. Therefore, the Q value can be considered as a function that maps each state-action pair into a real number $\Re$, which can be represented as

$$Q : S \times A \to \Re \tag{2.1}$$

The Q value is updated by observing its reward and resulting state when performing an action, which can also be considered as the expected value of the overall reward. For a given state s and action a, its Q value can be calculated as

$$Q(s, a) = \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma Z(s')) \tag{2.2}$$

where $Z(s')$ is the expected reward of state s', and γ is the discount factor that indicates how much weight is given for the future value.

If the resulting state s' is also not known, then the learning must be purely based on the state-action pairs (s, a), which is essential the Q-learning method. In Q-learning, the Q values are recursively updated from the old values:

$$Q'(s,a) = Q(s,a) + \alpha \cdot (R_a(s) + \gamma \max_a(s', a) - Q(s,a)) \tag{2.3}$$

where α is the learning rate controlling the weight of the new reward value. When $\alpha = 1$, the old value is totally ignored for each update. Note that α may not be a constant value. In fact, α must decrease with time to ensure the Q-learning process will converge eventually [62].

2.2 Q-learning-based Routing

As we already explained, Q-learning is a model-free reinforcement learning method which updates the policy through continuous observation of the rewards of all state-action pairs [58]. In the context of routing, we must first define the states and the actions. In fact, packet forwarding can be easily modeled as a MDP: one packet needs to travel to the destination through several intermediate nodes, and during each step, stochastic transmission failure results in an unchanged resulting state. Clearly, here a state is just the location of each packet while the action set is the choices of next hop for each node.

What is the learning agent? The most intuitive answer may be the transmission packets because they can make decisions for the next hop by observing the transmission results. However, this is clearly not possible because once a packet

reaches its destination, it is out of the system. Without knowledge accumulation, Q-learning cannot be carried out. If mobile nodes are considered as the learning agents, then there is no longer any state change because once a packet is forwarded to the next hop, it is passed to another learning agent. To appropriately apply Q-learning to routing, all the nodes have to be combined together to form multiple learning agents. Each learn agent is responsible for one specific destination, which results in three-dimensional Q values:

$$Q : S \times A \times D \rightarrow \Re \tag{2.4}$$

where D is the set of destinations. Therefore, the corresponding Q value of node i choosing an action a to reach destination d can be denoted as $Q_i(d, a)$. Since in the routing problem, the action a and state s are associated with a specific node, for convenience, we denote the node of the next hop as a and the current node as s. For example, $Q_i(d, j)$ means the Q value for node i taking node j as the next hop to reach the destination d.

All the Q values are distributively stored in each node as shown in Fig. 2.2. Since the state s is always the current node, each node only needs to store the Q values of destination-action pairs $D \times S$. In fact, to shrink the space usage further, since node i can only reach its neighbors Adj_i, only $Q_i(d, a) \forall a \in Adj_i$ are stored. The corresponding update function can be represented as

$$Q_i'(d, a) = Q_i(d, a) + \alpha \times \left[R_a(i) + \gamma \max_j Q_a(d, j) - Q_i(d, a) \right] \tag{2.5}$$

where $\max_j Q_a(d, j)$ is the best future value if the next hop a is chosen. Note that

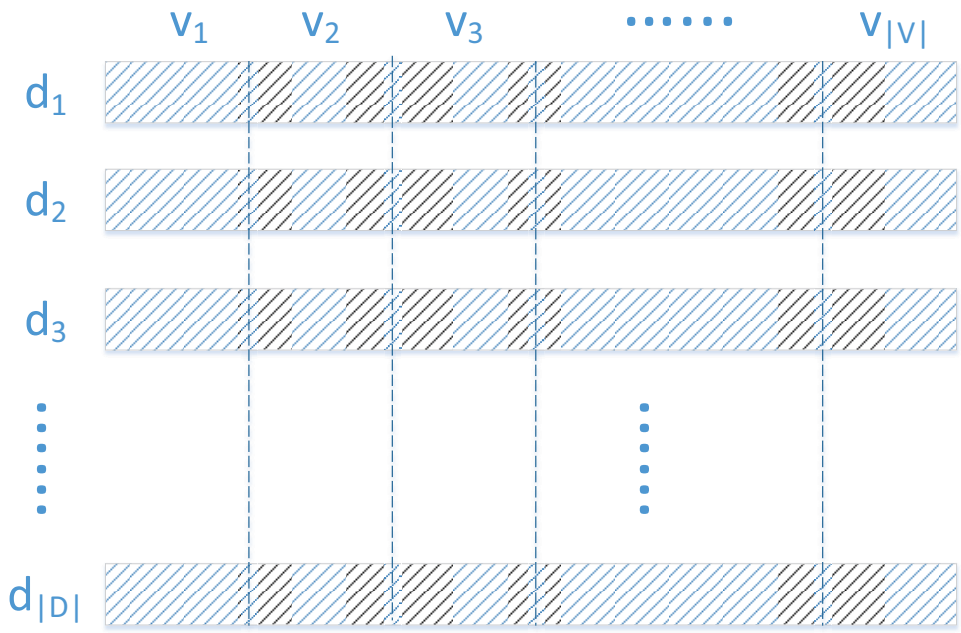


Fig. 2.2: Relationship between learning agents and nodes. Each row indicates a learning agent for a destination. All the learning agents are distributively stored in all the nodes

$\max_j Q_a(d, j)$ can only be retrieved from node a.

Another important decision for the Q-learning method is the choice of the reward function $R_a(s)$. Different reward functions indicate different preferences in the path selection process, just as in the maze example mentioned in the beginning of this chapter. It is difficult to argue which reward function is the best because the final performance is usually based on the application and the network characteristics. However, it is still possible to design a reward function which is of more robust and versatile applicability.

We choose $SINR$, *throughput*, and *delay* as the metrics to comprehensively indicate $R_a(s)$ value. $SINR$ is a common measurement for link qualities: higher $SINR$ usually results in more reliable links. *Throughput* is an important QoS parameter indicating the rate of successful message delivery, which is crucial for

real-time multimedia services. *Delay* measures the end-to-end latency of packet forwarding from the source to destination. High delay is likely to be caused by inefficient routing or severe congestion. Therefore, $R_a(s)$ is denoted as:

$$R_a(s) = -F(w_1 \cdot SINR)F(w_2 \cdot Throughput) \cdot w_3 \cdot Delay + Destination \quad (2.6)$$

where $SINR$, $Throughput$ and $Delay$ are the normalized feedback with their respective weights w_1, w_2 and w_3. $Destination$ is 1 only if the final destination is reached to encourage fewer hops. F is the discount function which maps the value into the range of $(0, 1]$. There are some interesting points about $R_a(s)$:

- $R_a(s)$ is always negative for all the nodes except the destination, which may have a positive reward. This is to ensure that no infinite routing loop is formed in the learning process.

- $Delay$ is treated differently because it is additive. For example, by adding the delay from node i to j and the one from node j to k, we can obtain the delay from i to k. $SINR$ and $Throughput$, on the other hand, do not have such a property. Therefore, in (2.6), they act as a discount factor to normalize $Delay$.

- A typical choice of F is $F(x) = e^{-x}$ because the quantity of an exponential function decays at a rate proportional to its current value. As a result, a fast drop when $SINR$ and $Throughput$ are small can be expected due to the minus sign in front.

- $SINR$, $Throughput$ and $Delay$ are all mapped into $[0, 1]$ through rescaling:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \quad (2.7)$$

Noted $SINR$ value can be recorded only when a packet is successfully received by the receiver. Therefore, $min(SINR)$ is the threshold $SINR$ value of the slowest supported transmission rate instead of 0. The detailed Min and Max values are shown below:

Table 2.1: QoS feedbacks

	Min	Max
$SINR$	SINR for minimal data rate	Transmission power divided by noise
$Throughput$	0	Raw data rate
$Delay$	Transmission time without contention	MAC transmission queue timeout value

- $SINR$ indicates the "buffer" to counter the mobility and channel fading, whereas $Throughput$ represents the link quality for a certain period in the past. Higher $Throughput$ indicates the quality and reliability of a neighbor. Unlike $SINR$ and $Delay$, which can be obtained at the moment when a packet is received, $Throughput$ has to be observed for a certain window by using a moving average function. For the same reason, it contains chronicle information about the link quality: it is expected that the more frequently used links are more reliable than those are solemnly used. For example, $Throughput = 0$ indicates there is no previous successful transmission attempts, which gives $F(w_2 \cdot Throughput) = 1$. In other words, no discount is given for the penalty $Delay$. As $Throughput$ gets higher, $Delay$ is discounted further.

The reward information is embedded in the CTS and ACK frames and send back to the sender. Unlike protocols described in [43] and [46], in which the reward is given only if the destination is reached, our protocol gives immediate reward

for both destination reached and good link quality. In [43] and [46], feedback only determines the discount factor γ. As a result, the hop count has a much higher priority than other factors.

So far we have shown how to store and update the Q values. The next step is to make the routing decisions based on the Q values. It may appear trivial that the Q entry with maximum Q value should be selected as the next hop. However, since the Q value can only be updated if the corresponding action is performed, a greedy strategy prevents the exploration process to find a better route. A simple solution is to normalize the Q values by the Boltzmann probability distribution [45] to enable exploration. The selection method can be represented as

$$P(R_i(d) = a) = \frac{e^{\beta Q_i(s,a)}}{\sum_{j \in Adj_i} e^{\beta Q_i(s,j)}} \tag{2.8}$$

where β controls the greediness of the selection process. When β is infinity, only the entry with the largest Q value will be chosen as the next hop. Whereas when β is 0, the choice becomes random in spite of the Q values. Note that QLR becomes a multipath routing scheme by introducing the Boltzmann-probability-based selection mechanism because traffic is divided to different intermediate nodes based on their Q values to mitigate the congestion level in a heavily loaded environment.

2.2.1 Information Sharing

As mentioned above, in order to update the Q values in (2.5), the maximal future value and reward have to be obtained from neighbors. They are shared using broadcasting and unicasting respectively:

- Each node broadcasts its maximal future values $\max_j Q_i(d, j) \forall d \in D$ by in-

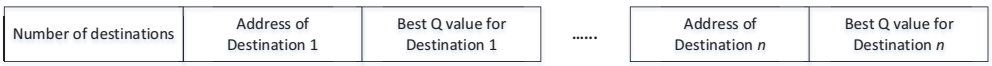


Fig. 2.3: The format of a Hello packet.

cluding the information into a Hello message as shown is Fig 2.3. Originally, the Hello message is periodically broadcast to confirm the adjacency relationships, which is an excellent candidate for the broadcasting information carrier.

- Each node unicasts the QoS information back to its neighbors. When a node receives packets from its neighbors, the SINR, throughput and delay of the sender is updated. By inserting it into the header of a routing control message, theses information is sent back to the sender whenever appropriate.

The information sharing mechanism inevitably introduces extra overhead. However, no extra packet is created during the process because the existing control packets are modified to accommodate the extra information. Moreover, to mitigate the corresponding penalty, it is best to enable the sharing mechanism at a fix interval. The guideline for choosing an appropriate interval is the network mobility: with higher mobility, higher information exchange frequency should be adopted.

2.2.2 Overview of Q-learning-based Routing

Generally speaking, the Q-learning-based routing process can be classified into the following phases:

- *Initialization:* In this phase, a node either just joins a network or the entire network just starts to communicate. Therefore, there is no Q entry available. If the node has packets to send, it needs to initiate a Route Request

(RREQ)/Route Replay (RREP) procedure to establish the first route. If the node has no packet to send, it will monitor HELLO messages from its neighbors. The corresponding Q entries of newly found destinations are initialized to 1 to encourage the initial explorations.

- *Learning:* In this phase, nodes have already obtained the list of destinations and initialized the corresponding Q entries. When it needs to send or forward a packet, all the Q values are retrieved to calculate the next hop. Once the packet is successfully received, the corresponding QoS feedback is embedded into an ACK packet and sent back by the receiver. Then, the node can update the corresponding Q values.

- *Sharing:* In this phase, nodes broadcast their optimal Q values for each destination to neighbors through the HELLO message. Meanwhile, once a HELLO message is received, nodes also update their neighbors Q values accordingly.

- *Maintenance:* This phase occurs when there is a change in the network. It can either be a topology change, such as nodes leaving the network, or a traffic change, such as extra flows emerging. A topology change can be observed from the HELLO messages. For example, when a HELLO message from a new node is received, the new node is added into the neighbor Q table. Whereas if the HELLO message from a neighbor is not heard for a certain timeout period, we can deduce the neighbor's absence. When a topology change is detected, the learning rate is reset to 1 to explore new paths. If a neighbor becomes unavailable, all the corresponding entries are removed from its neighbor's Q tables. When there is not entry left, the node enters the initialization phase.

A traffic change refers to a change of the destination set. When a node receives a RREQ request targeting itself, it adds itself into the HELLO message and broadcast. On the contrary, if after a timeout period, there is no more data packet received, this node will remove itself from the destination set by indicating a *N.A.* value in its optimal future value. Nodes receive such a value will remove the destination as well.

2.2.3 An Example of Q-learning-based Routing

Fig. 2.4 shows an example of how the Q-learning-based routing scheme works. Four nodes A, B, C and D are mobile nodes connected by wireless links. Source A tries to send packets to destination B, with two intermediate nodes C and D helping forward packets. For demonstration purpose, we assume that all the weights and parameters are set to 1.

Initially, similar to AODV, node A sends out a route request that floods through the network to reach node B. After B receives the request, it will identify itself as a destination and start to send back routing information. At first the routing information can only be sent from the destination B (assuming no other nodes are aware of B at the beginning). After the feedback from node B is received, node A, C and D will update their respective entries (Fig. 2.4a). Since node B is the destination, they can get the *destination* reward of 1 as in (2.6). For example, when C receives the feedback tuple as $(0.3, 0.3, 0.1)$, $R_C(B)$ is updated as 0.945. A, B and C then share such information with each other as shown in Fig. 2.4b. For example, A sends back its feedback to C indicating the reward and maximal future value. After that, C can update its entry as 0.539. Note that although the instant reward from A to C is the same as the one from C to A, it is not necessarily true for all links. The absence of the symmetric link assumption is also

the reason why we must adopt the feedback mechanism to obtain information.

Once a packet needs to be forwarded, the Q values are used to determine next hop according to (2.8). In this particular instance, A forwards packets to B, C and D with the probabilities of 31.77%, 29.98% and 38.26% respectively.

It is expected that after finite iteration, the Q values should converge to a fixed value in a static environment. In the next part, we are going to illustrate how to guarantee the convergence.

2.2.4 Parameter Tuning

There are quite a number of parameters in QLR, namely, the learning rate α, discount factor γ in (2.5), the Boltzmann probability parameter β in (2.8), and the three weights (w_1, w_2, w_3) in (2.6).

The convergence of Q-learning requires the learning rate to fulfill two conditions [62]:

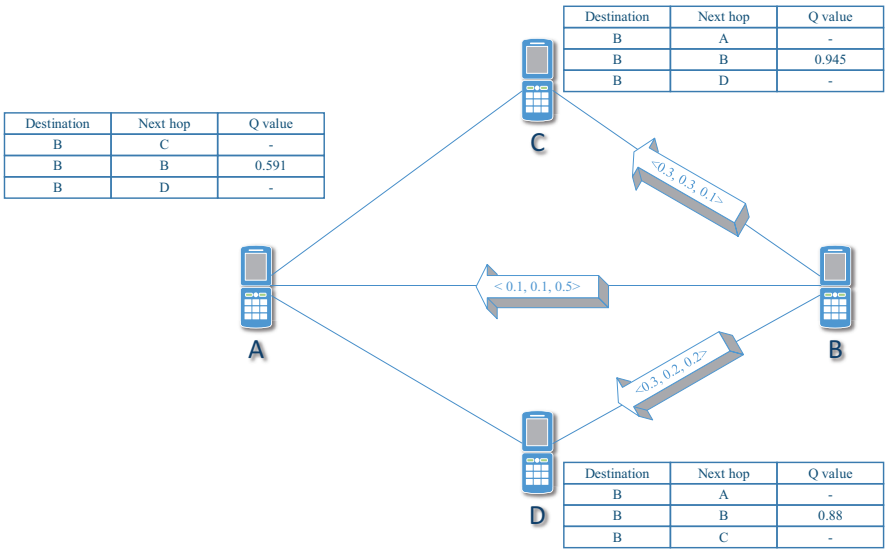
$$Condition\ 1 : \sum_t \alpha_t = \infty \tag{2.9}$$

$$Condition\ 2 : \sum_t {\alpha_t}^2 < \infty \tag{2.10}$$

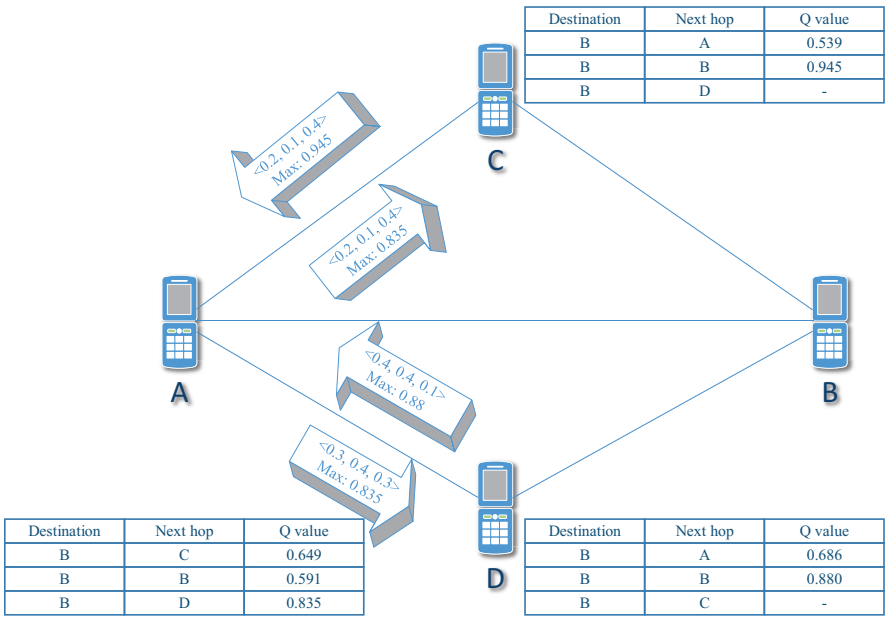
where α_t is the learning rate at tth iteration. t is initialized as one and increments by one whenever the Q value is updated. Note that a node maintains a different t for each destination. The first condition ensures that all values are reachable whereas condition 2 ensures that it will converge to an optimal solution.

The learning rate therefore can be defined as

$$\alpha_t = 1/t \tag{2.11}$$



(a) *At first, node B broadcast information to node C and D*



(b) *Node A, C and D exchange information and update Q values*

Fig. 2.4: QLR example. An example showing how the Q-learning-based routing protocol works

which meets both conditions 1 and 2 when $t \to \infty$. In other words, if the entire network remains static, the learning process is guaranteed to converge to a solution.

Unfortunately, neither $t \to \infty$ nor the static network assumption can hold in MANETs: we cannot expect a node to wait forever to find a path to the destination and all nodes just stay where they are throughout the learning process. One important feature of the Q-learning-based routing is its ability of providing best any-time solutions. In other words, although the solution provided may not be the optimal one, it is still the best solution so far can be found.

The Boltzmann distribution parameter β indicates the greediness of the actions. Therefore, it is desirable that a smaller value should be given for exploration at the beginning and becomes greater as more information is obtained. Therefore, we can also define β as

$$\beta_t = t \tag{2.12}$$

However, the above settings can only work as long as the network remains static. Once the topology changes, previous learning results will become obsolete. Therefore, a learning reset mechanism is required for continuous learning in a dynamic network environment like MANETs. Fortunately, since neighbors periodically send out Hello packets, it is quite easy for each node to keep a record of its neighbors. Whenever there is a change in the neighbor list, such as adding or deleting a neighbor, t will be reset to 1 to initiate the learning process. Note that when a topology change is detected, the t values for all the destinations are reset.

Weight tuning is inevitable for a reward function like (2.6), because all the

metrics are in different units. By introducing weight to each of them, we can then have a way to later consolidate their combined effects. Usually, weight tuning can be accomplished via heuristic approaches: explicitly try various combinations of weights and choose the one with the best performance. However, since different scenarios may require different weights, it is not possible to provide weights for each scenario. Therefore, we need to extract some common features from each scenario and group them accordingly. By doing so, only a limited set of weights are required for these groups.

The congestion level is a good measurement indicating surrounding traffic conditions. For a higher level of congestion, smaller weights are preferred because higher weights usually result in more hops, which makes network congestion more severe. The congestion level C can be defined as

$$C = \frac{\text{Channel busy time}}{\text{Total time}} \times \frac{\text{Current queue size}}{\text{Maximum queue size}} \tag{2.13}$$

We define the levels of congestion as light, moderate or heavy by defining two threshold value C_m and C_h such that the congestion level is light if $C < C_m$, moderate if $C_m < C \leq C_h$ and heavy elsewhere. For each congestion level, there is a corresponding heuristically tuned weight set, so that once the congestion level is estimated, the corresponding weight set will be chosen to calculate the reward.

The overall procedure for sending and receiving a packet is explained in Algorithm 1 and 2.

2.2.5 Routing-Loop Problem

The routing-loop problem is common in all the Q-learning-based routing protocols because of the explicit use of past experiences and neighboring information,

Algorithm 1 Procedure of packet sending

```
1: function SendPackets(packet, destination)
2:     nextHop ← GetNextHop(destination) using 2.8
3:     packet.SetNextHop(nextHop)
4:     if feedbackRecord.Contains(nextHop) then
5:         packet.InsertFeedback(feedbackRecord[nextHop])
6:     end if
7:     Send(packet)
8: end function
```

Algorithm 2 Procedure of packet receiving

```
1: function ReceivePackets(packet)
2:     if packet contains feedback then
3:         feedback ← Retrievefeedback(packet)
4:         w ← RetrieveWeight
5:         R_a^x ← CalculateReward(feedback, w)
6:         UpdateQEntry(R_a^x)
7:     end if
8:     RecordFeedback(packet)
9:     if packet.destination == self then
10:        Dispatch(packet)
11:    else
12:        SendPacket(packet, packet.destination)
13:    end if
14: end function
```

which, unfortunately, can easily be outdated in a dynamic environment.

We can take another look at the example above. This time, assume that node B dies due to insufficient energy. After a certain period, such a change will be observed by node A, B and C. They delete their corresponding entries which have node B as the next hop, and reset their iterate count t as shown in Fig. 2.5. However, the routing table of node A still contains entries that can reach node B. In other words, node A still thinks node B is reachable through node C or D. At the same time, node C and D also believe that they can reach node B with the help of node A. That is the moment a routing loop is formed when any of these nodes tries to send packet to node B.

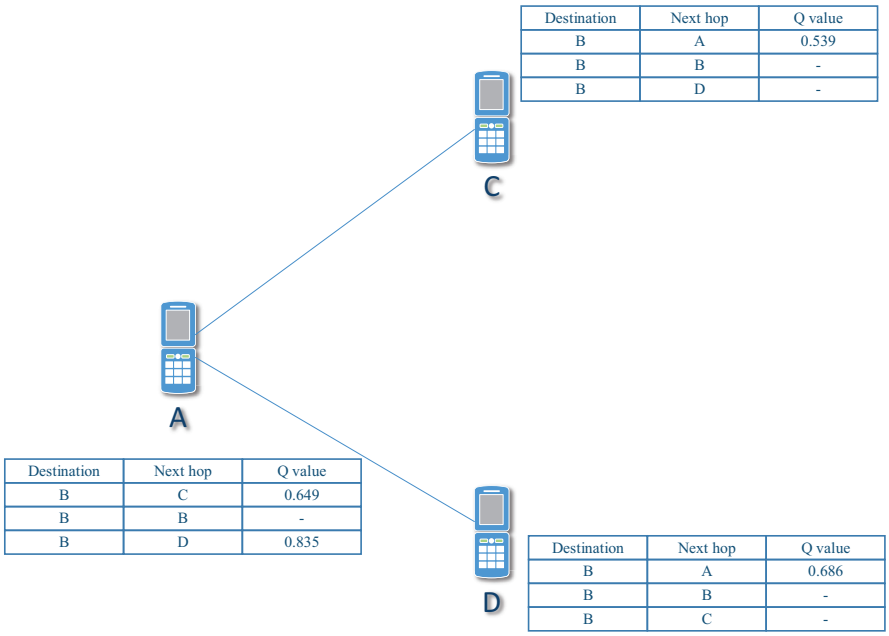


Fig. 2.5: An example of the routing-loop problem. If Node B goes down, all the nodes will delete the entries with node B as the next hop and reset t. However, such a change may result in a routing loop because node A and C still think they can reach node B through each other.

Probing packets can be used to solve the routing loop problem. Every fixed time interval, typically 5 seconds, a source node sends out a probing packet to its destinations. This probing packet records every node it traverses. Therefore,

whenever an intermediate node receives the same packet twice, a routing loop is detected. It will delete the corresponding routing entries in this loop, and then the probing packet will be forwarded until all the nodes in this loop are notified.

Taking Fig. 2.5 as an example, if node A detects that node A and C are forming a routing loop, it will delete the entry $Q_A(B, C)$. Then in the next 5 seconds it will delete $Q_A(B, D)$ once $A - D$ is found to be a loop.

The detailed procedure for the loop detection is shown in Algorithm 3.

Algorithm 3 Procedure of receiving a probing packet

```
1: function ReceiveProbingPackets(packet)
2:     if packet is received more than twice then
3:         discard packet
4:     else if packet is received twice then
5:         A loop is detected, extract loop information from the packet and delete
   corresponding entries
6:         Forward packet
7:     else
8:         Add current node information into the packet
9:         Forward packet
10:    end if
11: end function
```

2.2.6 System Analysis

It is challenging to combine all the nodes in a network as learning agents when they can join and leave the network at any time. Even so, there is no need for a higher level root supervisory system. In fact, the reason we choose Q-learning is because of its capability of handling node mobility in MANETs.

Although all nodes in the network form a learning agent for a given destination, it is not necessary that one transmission involves all the nodes. Therefore, when one node leaves the network, only part of the learning agent is reset, which does not affect the rest of the network. More importantly, the Q-learning-based

routing protocol enables the multi-path routing feature. In other words, multiple paths are stored in each node. When one path fails to function, there are alternative routes available to ensure connectivity.

When a node joins the network, it will monitor the broadcast packets to obtain knowledge of the surrounding environment and figure out which destinations can be reached. Then, it broadcasts such information to its neighbors. The neighbors then update their own routing entries or add new routing paths. More details can be found in the example below.

Another important consideration is the convergence rate, which reflects nodes' response time to a network change. For example, once a node leaves or joins a network, we need to know how much time is needed for itself and rest of the network to recognize the change and adapt to it accordingly. We have tested the convergence time for a newly added node in different scenarios with the number of neighbors changed from 2 to 20. The data rate is set to 10 packets / second. The result of 1000 tests is shown in Fig. 2.6. It can be seen that when the number of neighbors are 20, the learning process can still finish in about 6 seconds from an initial state. Note that the time required to find the best path is less than the convergence time because better routes have more chances to be visited. It is therefore updated faster than the rest.

2.3 Experimental Results and Discussion

2.3.1 Simulation Setup

Simulations are carried out by using Network Simulator 3 version 3.12.1 (ns-3.12.1) [63] in Ubuntu [64] version 11.04. The network consists of multiple mobile nodes with Wi-Fi communication devices equipped. We assume that the IEEE

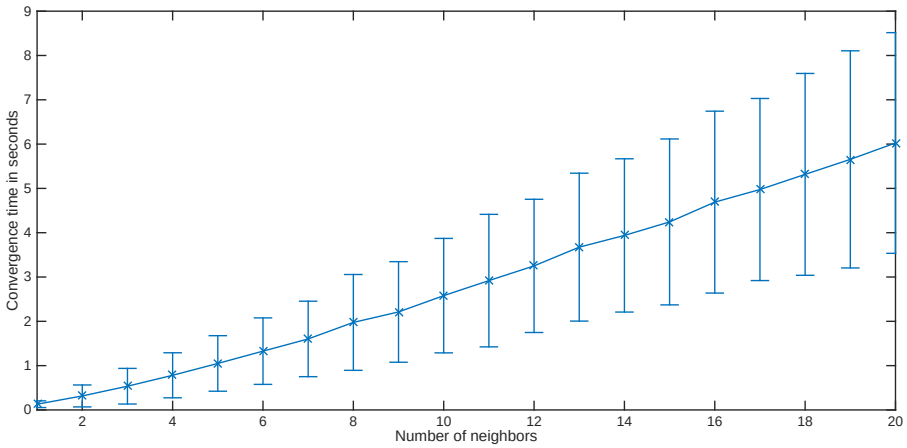


Fig. 2.6: Convergence time for different number of neighbors

802.11b DCF mode is used with RTS/CTS enabled.

AODV [36] and QLMAODV [46] are chosen as the benchmark to evaluate the performances. AODV is a reactive minimum-hop-based routing protocol for MANETs. The route discovery process is initiated by the source, which broadcasts a requests packets to it neighbors. Other node who receives the request packet, re-broadcast it if it is not destination and does not know a path to the destination. Otherwise, it will reply the request. Note that the replied packet does not contains information about the full path, therefore, each node is only aware of the next hop for each destination.

QLMAODV is a Q-learning-based routing protocol which uses the discount factor to differentiate the optimality of its neighbors. The discount factor can be expressed as $\gamma = MF * BF * EF$, where MF, BF and EF is the abbreviation of mobility factor, bandwidth factor and energy factor, respectively.

We simulate a dense MANET consisting of up to 100 nodes uniformly distributed in a 300 m $\times$ 1500 m environment. Flows are generated using random source-destination pairs. One node can be both source and destination. However,

Table 2.2: Simulation Parameters for Routing

MAC Layer	IEEE 802.11 DCF with RTS/CTS
Simulation area	1500 m × 300 m
Mobility pattern	Random way-point model
Maximum speed	2-10 m/s
Traffic flow	Constant Bit Rate (CBR)
Packet size	512 bytes
Flow rate	10 packets/sec
Flow number	5-40
Raw stream data rate	2 Mbps
Simulation time	400 seconds or longer
Large-scale propagation model	Log-distance path loss model, $n_l = 3.0$
Fast fading model	Ricean fading, K = 13 dB

the source of a flow cannot be the source of another flow. The same rule applies to the choice of destinations as well. During the simulation, we assume that all nodes are cooperative: whenever there is an incoming packet, a node tries to forward it to the destination in spite of its current conditions. In other words, no packets are intentionally dropped. The summary of simulation parameters is shown in 2.2.

Their performances are evaluated under different node densities, traffic loads, mobility and link qualities. Each measurement is an average of 60 runs with different random number seeds. By constantly observing the results, each simulation terminates at 400 seconds or the time when two adjacent observations are within 5% differences, whichever is longer, to ensure convergence. The results are shown with 95% confident interval.

2.3.2 Simulation Metrics

The following end-to-end QoS metrics are chosen to evaluate the performances:

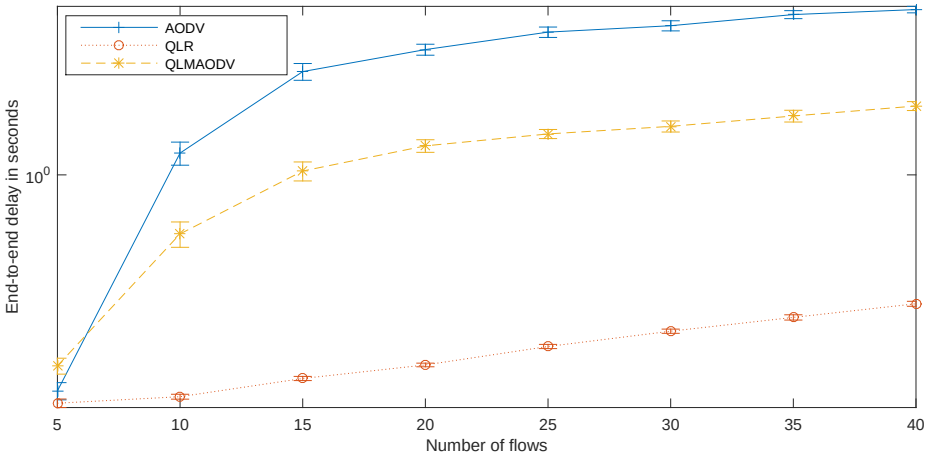
- Packet Delivery Ratio (PDR): The ratio of the number of data packets received by the destination to the number of data packets transmitted by the source. Since constant rate is applied to all transmissions, PDR measurement is indicative of effective throughput.
- Delay: The average duration between the moment a data packet is sent and the moment it is received by the destination. If a packet is retransmitted, the sending time is recorded from the moment of its first attempt.

The simulation performance data are collected by the NS3 flow monitors.

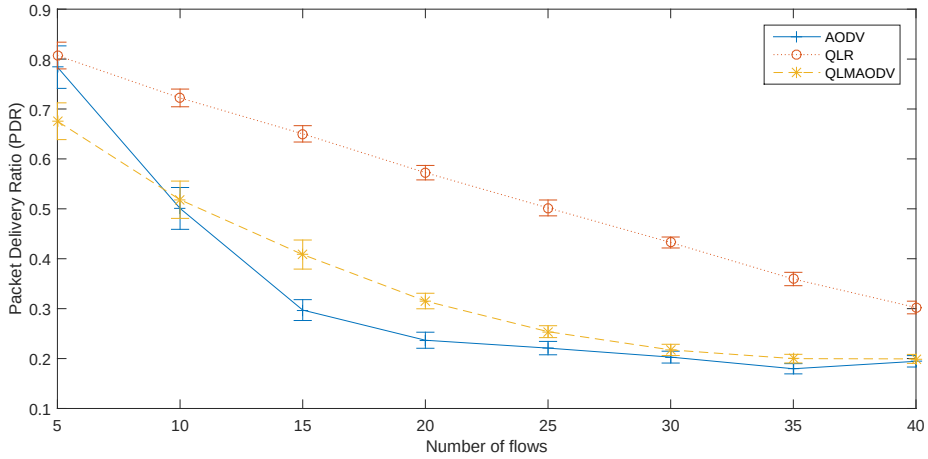
2.3.3 Traffic Load

In this scenario, the number of nodes is capped at 50 while the number of flows increases from 5 to 40. We stop the measurement at 40 flows because the PDR values at 40 flows are close to 0. The traffic of each flow is generated at 10 packets per second. The delay and PDR results are shown in Fig. 2.7.

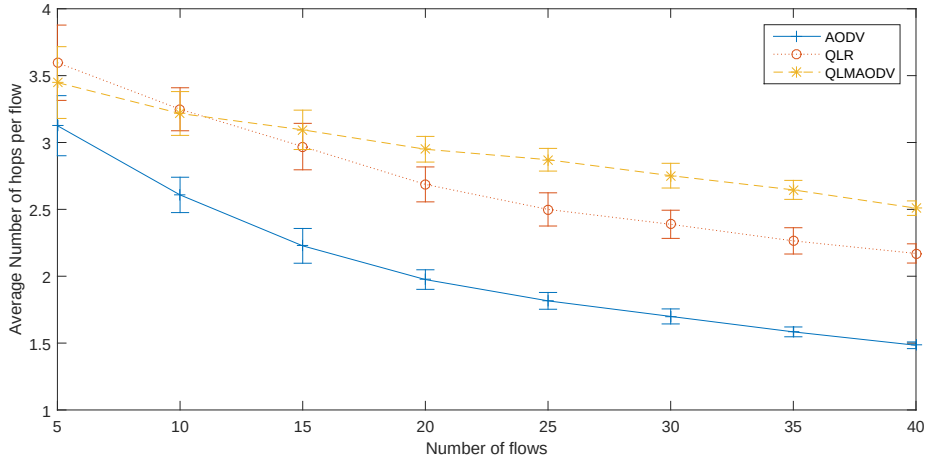
Fig. 2.7a shows that AODV is susceptible to heavy traffic. The delay increases dramatically when the number of flows increases from 5 to 10. This is mainly due to the shortest path algorithm adopted by AODV, which can easily create congestions in dense areas. In order to reduce the communication overhead, AODV prefers to use the existing paths for packet forwarding. As a result, nodes in dense areas can easily be overloaded. QLR and QLMAODV, on the other hand, try to avoid such areas by seeking better QoS feedback. Thanks to the weight tuning mechanisms, when traffic load gets higher, QLR outperforms QLMAODV.



(a) Delay



(b) PDR



(c) Number of hops

Fig. 2.7: QoS performance under different traffic loads

When the congestion level becomes severe, a more direct path is preferred because traversing more hops may eventually worsen the overall congestion level. Although AODV chooses the most direct paths, they are shared among many flows, which results in severe performance impairment.

PDR in Fig. 2.7b also shows the same pattern. When the number of flows reaches 10, both AODV and QLMAODV suffer a great loss in PDR due to congestion. Although the bandwidth factor is explicitly taken into consideration, QLMAODV does not perform well because congestion is not necessarily caused by the lack of bandwidth, it can also be caused by excessive contentions. In the latter case, QLMAODV still tries to send packets to those congested intermediate nodes which results in more packet loss. Moreover, the lack of a weight tuning mechanism makes it hard to differentiate the factors. For example, in a congested network, the mobility and energy factor have less impacts on the final performance, therefore should be given smaller weights. This is also the reason we introduce the weight tuning mechanism to help the nodes adjusting their learning behaviors. AODV has a good performance when the number of flows is 5 because short paths usually perform well when congestion is not formed. QLR has the best performance in PDR. It is expected that when the traffic load gets even higher, all of the three will have equal performance since without reliable information exchange, the Q-learning-based protocols cannot function.

Fig. 2.7c shows the average number of hops per flow for each protocol. Note that only successfully received packets are taken into account. It can be seen that when the network becomes congested, the average hops for AODV drops accordingly because packets taking longer paths get lost more easily during forwarding. The weight tuning mechanism adopted by QLR effectively shortens the transmission paths to minimize the congestion level. Although compared with

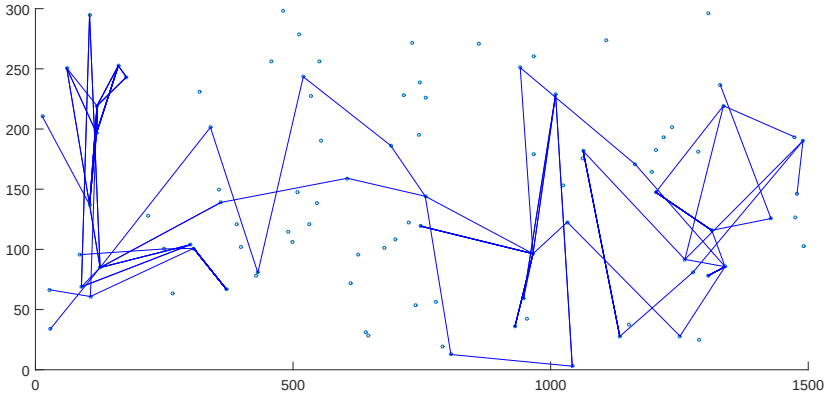
QLMAODV, it takes more hops to forward packets when the number of flow is 5. The relationship reverses soon after the number of flows reaches 10 and the difference increases with the traffic load.

Fig. 2.8 shows all the routing paths for the three routing protocols, when the number of flows is 20, at a specific time instance $t = 200s$. Each solid line indicates a specific link used as a part of a routing path. AODV (Fig. 2.8a) uses very limited number of links, especially in the dense areas. Besides, since route rediscovery takes place only when current route breaks, there is no guarantee that the current path is the shortest in a dynamic network. QLMAODV (Fig. 2.8b) on the other hand aggressively chooses those intermediate nodes with good QoS feedback. As a result, routing paths spread all over the network but are not well balanced, as we can easily spot some "hot zones" where a lot of packet forwarding takes place. QLR forces nodes to take biased decisions based on different congestion levels. Therefore, it can be seen from Fig. 2.8c that routes are evenly distributed among the whole network which results in a more satisfactory QoS performance.

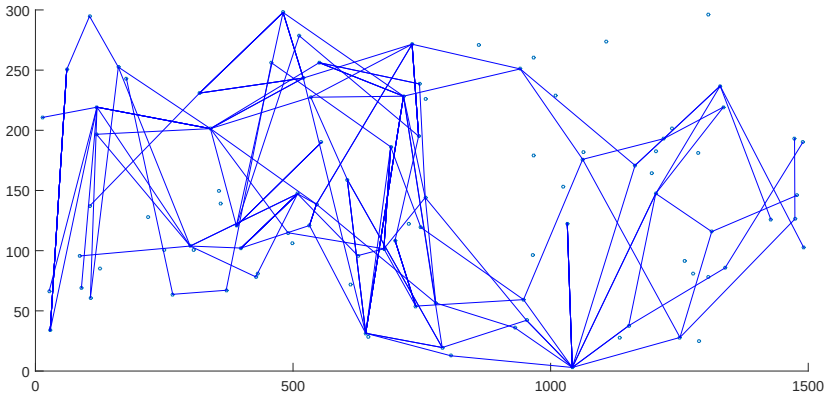
2.3.4 Node Density

In this scenario, the number of flows is fixed at 5 while the number of nodes is varied from 50 to 100. The QoS performance of the three protocols are shown in Fig. 2.9.

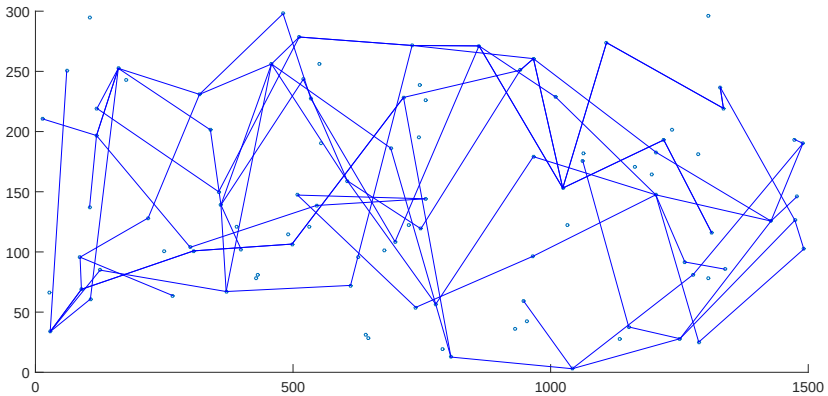
Generally speaking, higher node density means more neighbors around each node. Therefore, it inevitably causes more communication overhead, which may result in more severe congestions. On the other hand, higher node density also increases the network connectivity so that each node has more choices for packet forwarding. Therefore, the performance under high node density depends on two factors. One is the efficiency of the information sharing mechanism, which deter-



(a) AODV

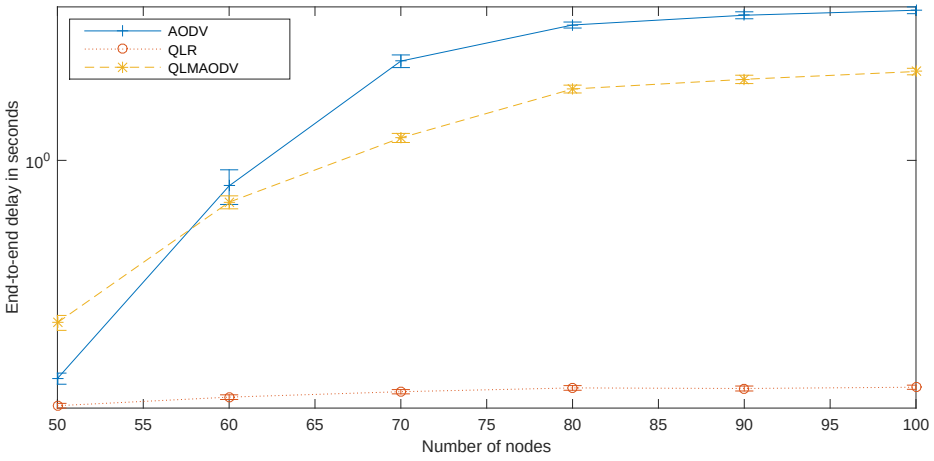


(b) QLMAODV

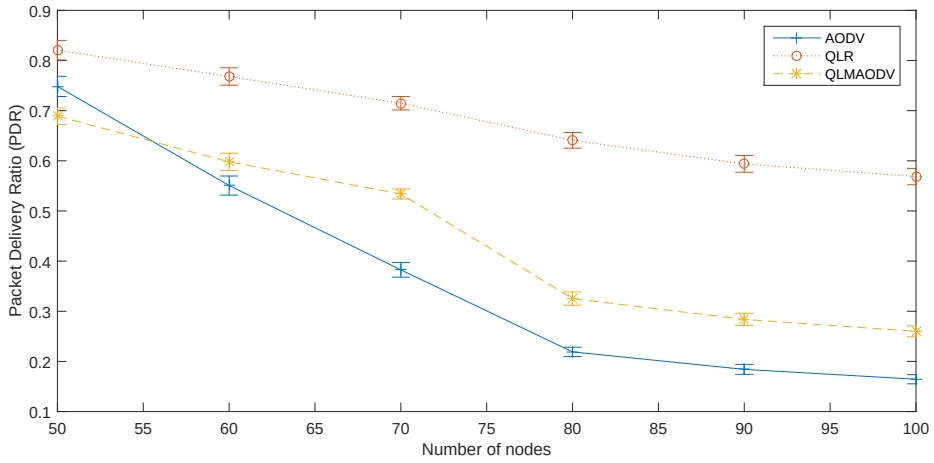


(c) Q-learning-based routing

Fig. 2.8: Routing path exmaple



(a) Delay



(b) PDR

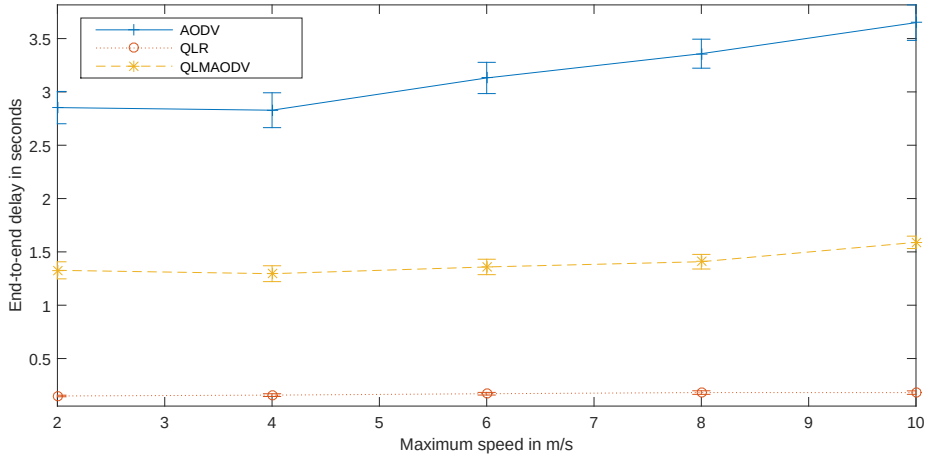
Fig. 2.9: QoS performance under different node densities

mines the size of the overhead introduced. The other is how these protocols take advantage of the extra routing options to mitigate the negative overhead effect.

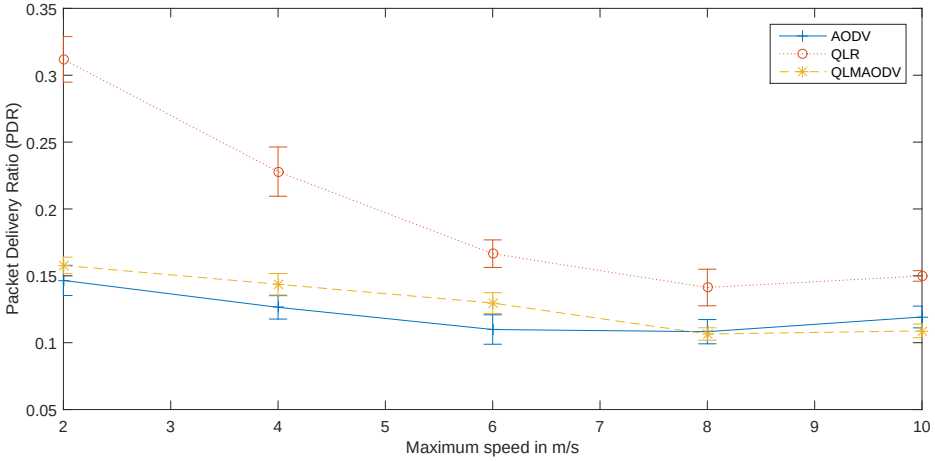
It can be seen that AODV is vulnerable to the increased node density as shown in Fig. 2.9. AODV sends out route requests only when the current path breaks, therefore the merit of having more neighbors is not significant. On the contrary, it is easier to find a neighbor at the edge of its transmission range, which usually has a poor link quality. The shortest path algorithm may choose those nodes as the next hop which results in poor performances. QLMAODV faces a different problem in this scenario: the routing loop problem can exhaust the network resources very quickly. As the node density increases, the probability of forming a routing loop also increases. QLR has the best performance in this scenario. However, if we compared it with the performance in scenario 1, we can find that the traffic load has more significant impacts on the delay performance. The information embedded in the Hello message is proportional to the number of destinations. Therefore, when the number of flows increases, the information sharing overhead increases proportionally. QLMAODV suffers a serious PDR drop of more than 20% when the number of nodes increases from 70 to 80 (Fig. 2.9b). The routing decisions of QLMAODV are based on the broadcast information. When the PDR value drops below 50%, the reception of the information becomes no longer reliable, which further reduces the PDR value.

2.3.5 Mobility

In this scenario, we examine the effect of frequent topology changes on the final QoS performances. The number of flows is fixed at 40 whereas the maximum speed of the random way-point model is varied from 2 m/s to 10 m/s. The simulation results are shown in Fig. 2.10.



(a) Delay



(b) PDR

Fig. 2.10: QoS performance under different mobilities

The impacts of mobility on the final QoS performance are limited for AODV and QLMAODV because of their route maintaining mechanisms. However, the learning process of QLR relies on topology of neighbors. If there is a topology change, QLR will reset t to initialize the learning process. As a result, high node mobility prevents QLR from converge. This is also the reason in Fig. 2.10b, PDR of the QLR drops from 31.6% to 15.1%.

2.3.6 Link Quality

Link quality directly determines the stability of a transmission. Higher link quality can ensure higher PDR and smaller route re-establish cost. This scenario evaluates the effect of link quality by varying the factor K of the Ricean propagation model, whereas fixing the number of nodes and flows to 50 and 20 receptively. K is the ratio between the power in the direct path to the cumulative power of indirect paths. Therefore, a smaller K value means a smaller portion of line-of-sight signal, and therefore less stable transmission links. By observing the final delay and throughput performances, we can evaluate how these three protocols handle frequent link breaks. The simulation results are shown in Fig. 2.11.

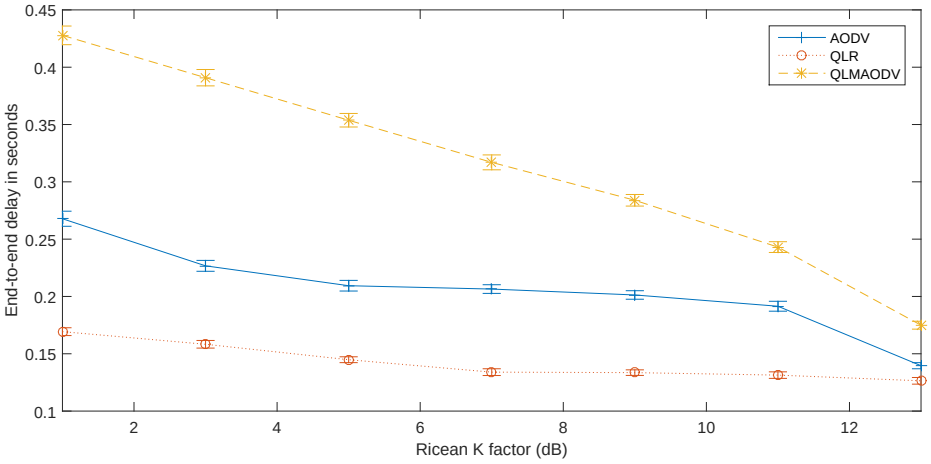
As shown in Fig. 2.11a, QLR is less vulnerable to unstable links because SINR is explicitly taken into account as a metric in the reward function. As a result, the average SINR can be obtained through continuous learning process. QLMAODV has the longest delay because its cost function does not consider the channel fading effect. Therefore, although nodes with larger bandwidth may have a worse performance due to frequent packet loss, QLMAODV still prefers to choose them as the next hop. In fact, this is exactly the reason why QLMAODV has the lowest PDR as in Fig. 2.11b.

The performance of AODV is fair. It has a longer delay than QLR because it

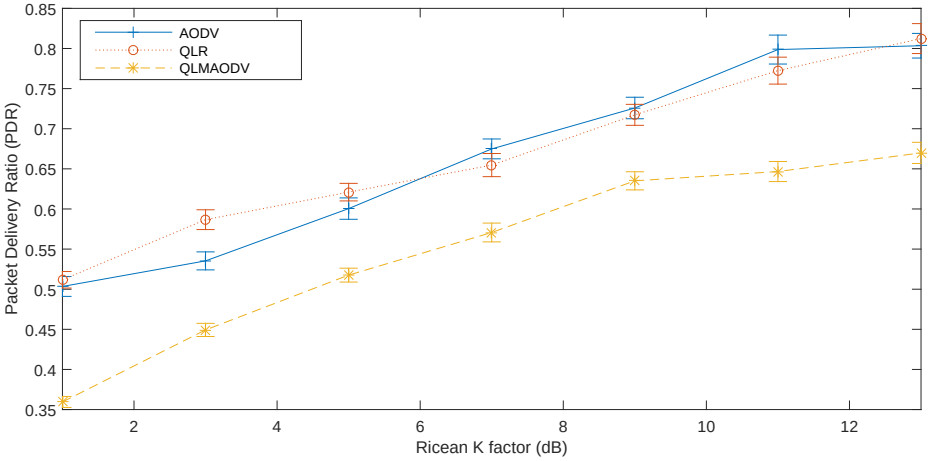
starts the route repair process when links break, which takes extra time to establish a new route. QLR, on the other hand, chooses a secondary route when the first route fails without introducing any extra route request. It is interesting to notice that although AODV does not consider link qualities, the PRD performance is comparable with QLR (Fig. 2.11b). During the route discovery process of AODV, the source sends out a Route Request (RREQ) packet which floods through the network until the destination is reached. Then the destination replies back a Route Reply (RREP) message following exactly the same path. Therefore, the RREQ and RREP packets have to survive all the links along the path between the source and destination, to establish a route. In other words, the route has been tested for reliability when it is established. It can also be seen that the effects of fading are much more pronounced than those of distance attenuation [65].

2.4 Summary

In this chapter, we give a short introduction of the reinforcement learning and Q-learning method, followed by a detailed explanation about how they can be applied to the MANET routing problem. Then we optimize the learning process by introducing the weight tuning, parameter reset and routing-loop prevention mechanisms. Finally, we compare QLR with the benchmarks by simulations. The results show that QLR can perform well in different traffic loads, node densities, and link qualities. However, the parameter reset mechanism is heavily affected by the node mobility.



(a) Delay



(b) PDR

Fig. 2.11: QoS performance under different Ricean K factors

Chapter 3

Power-controlled Routing Protocol

The idea of putting a power control scheme into a routing protocol is straightforward because power control and routing are closely related to each other. It is illustrated in chapter 1 that by constructing a topology based on the routing requirements, the overall performances can be improved. However, it is not easy to find the optimal power level. The transmission range is directly determined by a node's transmission power. Therefore, a higher transmission power level can provide higher connectivity and shorter path. On the other hand, a larger transmission range causes more interference to the neighbors and may further impair the overall network performance. Trade-offs between transmission range and interference level is of paramount importance in power-controlled routing protocol design.

In chapter 2, we have shown how the Q-learning method can be applied to the routing problem. In the following content, we are going to present the Q-

Learning-based Power-Controlled Routing (QLPCR) scheme by explaining how power control can be added into the Q-learning-based framework. Besides, a routing-loop prevention mechanism is proposed based on an improved information sharing system. The simulation results show that our method can achieve better performance under different loads, mobility levels, and node densities.

3.1 System Design

3.1.1 Motivation

Since the Q-learning method is based on a black-box model, we can simply extend the action space A and the instant reward $R_a(s, s')$ by adding power control into consideration. Extending the action space A to include power control is straightforward: we can transform A into a two-dimensional action space, i.e. $A = A_r \times A_p$ where A_r and A_p are the action space of routing and power control, respectively. Therefore, the Q value can be extended correspondingly:

$$Q : S \times A_r \times A_p \times D \to \Re \tag{3.1}$$

Note that the original Q-learning algorithm requires the actual feedback/reward from neighbors for each state-action pair during the discovery phase. It soon becomes impractical if the action space gets larger. Explicitly trying each action may never reach a convergence point because of the constant changes of topology in MANETs. For example, let us consider a network consisting of 50 nodes and 10 traffic flows, each node has 5 different power levels. According to (3.1), there are totally $50 \times 50 \times 10 \times 5 = 12500$ Q entries in the network, which are distributively

stored in each node. Assume the average number of neighbors is 5, then each node has to maintain $5 \times 5 = 250$ Q entries with $5 \times 5 = 25$ available actions. In other words, a node has to explicitly try all the 25 actions and wait for the feedback until all the 250 Q entries converge.

It is not easy to extend the instant reward $R_a(s, s')$ as well. Recall that in chapter 2, we combine several QoS metrics to form the reward function. Therefore, a trivial solution is to add one more metric indicating the optimality of a power level. However, it is quite challenging because of the complicated interactions between power control and routing. Since the change of power level has impacts on multiples nodes, it is hard to describe such impacts in a single Q value. More importantly, such an approach makes the trade-off between transmission range and interference level unmeasurable.

Now it is clear that the trivial extension from chapter 2 is out of the question. Therefore, we need to redesign the whole system to provide solutions to the following problems:

- The action space is too large to be explicitly tried out.
- The optimality of a power level cannot be assessed directly.
- It is hard to measure the trade-offs between the transmission range and interference level.

Our solution is to define $R_a(s, s')$ as the end-to-end delay from node s to s'. To address the first problem, we need to break the black-box model of the Q-learning method. With the help of a CSMA/CA delay model, all the transmission delay can be estimated accordingly. We also define a new term "weighted delay", which is the product of transmission data rate and delay. The weighted delay is used to measure the trade-offs between transmission range and interference level. The

philosophy behind the weight delay is that it is better to leave the nodes with high data rate alone because interfering them may delay a lot of packets. On the other hand, a node with high data rate is given higher priority to use a larger power level if its performances can be improved. Note that by specifying the instant reward $R_a(s, s')$ as the delay to the next hop, the Q values now represents the end-to-end delay from the current node to the destination.

Therefore, the first step is to define a CSMA/CA model to calculate the end-to-end delay.

3.1.2 CSMA/CA Delay Model

The IEEE 802.11 DCF MAC protocol is based on CSMA/CA which is well known for its capability of mitigating the hidden node problem. We have already mentioned a lot of terms in CSMA/CA such as RTS, CTS and ACK etc., here we will give a brief introduction of CSMA/CA.

The CSMA/CA protocol continuously senses the carrier to avoid potential collisions. The flow chart in Fig. 3.1 indicates the key procedures in CSMA/CA. Initially, nodes with packets waiting to be transmitted listen to the shared medium until it becomes idle. Instead of transmitting the packet immediately, nodes decrement its backoff counter. This process repeats until the backoff counter drops to zero. Then, the RTS and CTS frames are exchanged between the sender and receiver to initiate the real data transmission process. Any node receiving the RTS or CTS remains silent for a period indicated in the Network Allocation Vector (NAV) as if the channel is busy. This process is known as the virtual carrier sensing, which is important in solving the hidden terminal problem. In case of a transmission collision, the sender doubles its contention window (CW) and sets the backoff counter uniformly between 1 and CW, which is known as the binary

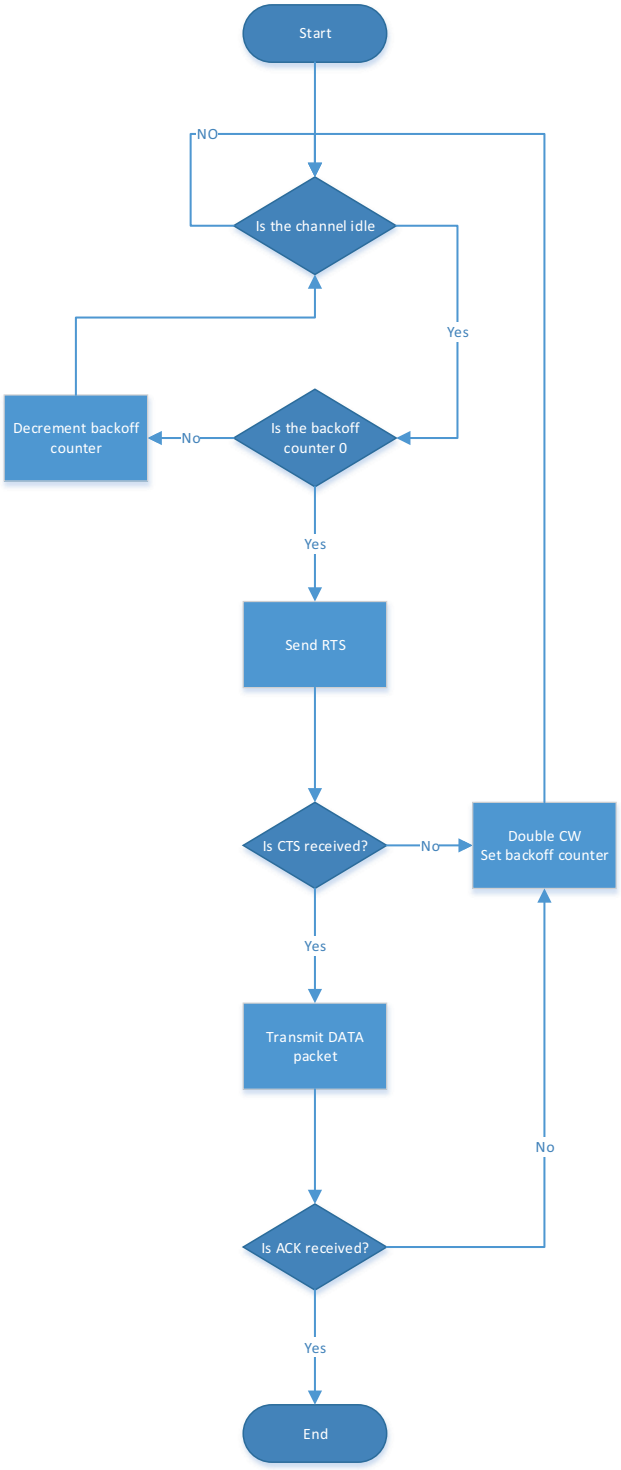


Fig. 3.1: Flowchart of CSMA/CA

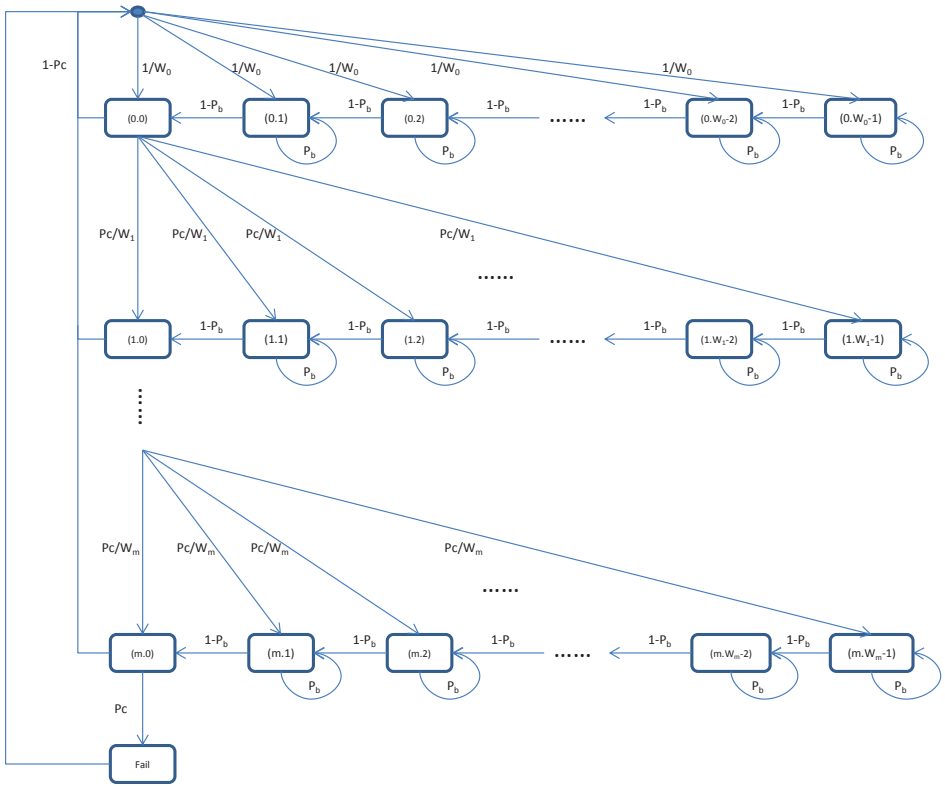


Fig. 3.2: Markov Chain model for backoff process

exponential backoff. Once the number of collision exceeds the maximal retransmission attempts allowed m, the packet is dropped. After a transmission finishes (both successfully and unsuccessfully), the backoff counter is reset to the minimal value $CW_{\min}$. The receiver sends back an ACK packet to the sender once the data frame is successfully received. If the sender does not receives the ACK frame within a certain time period, the transmission is considered failed and a retransmission process is initiated. This is also the reason why transmissions can succeed from node i to j only if the node $j \in adj_i$ so that the RTS, CTS, ACK frames can be exchanged.

A lot of research has been done on the CSMA/CA models [66–69]. Therefore we choose similar settings: a network consists of v contending nodes. All the nodes

share the same medium therefore at most one node is able to successfully transmit at any time slot. We also assume the perfect reception, i.e. if no collision happens during one transmission attempt, the packet is successfully received. The whole backoff process can be modeled as a Markov Chain model, which is a memoryless random process such that the probability distribution of the next state depends only on the current state [70]. The random process of the backoff counter matches the characteristics of a Markov chain model by defining each state as a pair of the retransmission count and backoff count (Fig. 3.2), with CW_i indicating the contention window for the ith retransmission, P_b and P_c representing the probability of channel busy and collision respectively.

Unlike [67, 68] which assume that each packet has an infinite number of retransmission attempts until it succeeds, the model is changed to have m maximal allowed retransmission attempts and compulsory backoff for a new packet transmission. Let $b_{i,k}$ is the state probability of $\{i, k\}$ where i is the retransmission state and k is the backoff counter state. Corespondingly, b_{fail} is the state probability of transmission failure. The following equations can be obtained from the Markov chain model:

$$b_{i,0} = {P_c}^i b_{0,0} \text{ for } 0 \leq i \leq m \tag{3.2}$$

$$b_{fail} = {P_c}^{m+1} b_{0,0} \tag{3.3}$$

$$b_{i,k} = \frac{CW_i - k}{CW_i} \times \frac{1}{1 - P_b} \times b_{i,0} \text{ for } 0 \leq i \leq m, 0 < k < CW_i \tag{3.4}$$

$$\sum_i \sum_k b_{i,k} + b_{fail} = 1 \tag{3.5}$$

where CW_i is the contention window for i^{th} retransmission and P_c is the probability of a transmitting frame colliding given that it is transmitted.

If we define τ as the average attempts per time slot for each node, then

$$\tau = \sum_i b_{i,0} = \sum_i {P_c}^i b_{0,0} = \frac{1 - {P_c}^{m+1}}{1 - P_c} b_{0,0} \tag{3.6}$$

P_b is the probability of sensing the channel busy which is the probability that among the remaining $v - 1$ nodes, one or more nodes transmit at one time slot duration. Therefore it is also the probability that another one or more nodes transmit at one time slot duration:

$$P_c = P_b = 1 - (1 - \tau)^{v-1} \tag{3.7}$$

From (3.2) - (3.7), $b_{0,0}$, τ, P_c and P_b can be worked out. P_s is defined as the probability that one transmission attempt is successful, which can be calculated as the probability that only one node transmits given at least one node is transmitting:

$$P_s = \frac{v\tau(1 - \tau)^{v-1}}{1 - (1 - \tau)^v} \tag{3.8}$$

Since the collision probability for each transmission attempt is P_c, we can derive that a transmission succeeds after nth attempts with the probability of $(1-P_c){P_c}^{(n-1)}$ and fails with the probability of ${P_c}^m$. Note that in CSMA/CA, if a node senses channel busy, its backoff counter will pause until the medium is free again. Therefore, we can first define X as the backoff delay without the counter pause time, which is measured in terms of number of time slots.

The probability that the nth transmission is successful is $(1-P_c){P_c}^{n-1}$ with the corresponding expectation of backoff time is $\frac{CW_{\min}+1}{2}\sum_{i=1}^{n}2^{i-1} = \frac{CW_{\min}+1}{2}(2^n-1)$. If all the m transmission attempts fail, the backoff time is $\frac{CW_{\min}+1}{2}(2^m-1)$ with the probability of ${P_c}^m$. Therefore, when $P_c \neq 0.5$, the expectation of X can be written as

$$
\begin{aligned}
E[X] &= \frac{CW_{\min}+1}{2}[(1-P_c)\sum_{n=1}^{m}{P_c}^{n-1}(2^n-1)+{P_c}^m(2^m-1)] \\
&= \frac{CW_{\min}+1}{2}[\frac{1-P_c}{P_c}\sum_{n=1}^{m}{P_c}^n(2^n-1)+{P_c}^m(2^m-1)] \\
&= \frac{CW_{\min}+1}{2}[\frac{1-P_c}{P_c}(\sum_{n=1}^{m}(2P_c)^n-\sum_{n=1}^{m}{P_c}^n)+{P_c}^m(2^m-1)] \\
&= \frac{1}{2}(CW_{min}+1)\left((2^m-1)\,P_c^m+\frac{(1-P_c)\,(2\,(2^m-1)\,P_c^{m+1}-(2^{m+1}-1)\,P_c^m+1)}{(P_c-1)(2P_c-1)}\right) \\
&= \frac{(CW_{min}+1)\,(2^mP_c^m-1)}{4P_c-2}
\end{aligned}
$$

On the other hand, when $P_c = 0.5$, $E[X]$ can be expressed as

$$E[X] = \frac{(CW_{min}+1)m}{2}$$

Therefore, by combing them together, we can have

$$E[X] = \begin{cases} \frac{(CW_{min}+1)(2^m P_c^m - 1)}{4P_c - 2} & P_c \neq 0.5 \\ \frac{(CW_{min}+1)m}{2} & P_c = 0.5 \end{cases} \tag{3.9}$$

Then, we can add the backoff counter pause time into the total backoff delay B. The expectation of B can be calculated as

$$E[B] = E[X] + E[X]P_b(P_s T_s + (1 - P_s)T_c) \tag{3.10}$$

where T_s and T_c are the average time that the channel is occupied with a successful and collided transmission, respectively. P_b is the probability that at least one of the neighbors transmits at each time slot. Therefore, the average transmission made by the neighbors in X slots is $E[X]P_b$. $P_s T_s + (1 - P_s)T_c$ calculates the average time for each transmission attempt.

T_c is relatively simple: after the sender sends out the RTS packet and detects the collision, it will wait for another Extended InterFrame Space (EIFS) period. A successful transmission consists of exchanging control messages and real data transmissions. There is one Short InterFrame Space (SIFS) buffer period between two adjacent control frames and one DCF InterFrame Space (DIFS) at the end of the transmission. Therefore, T_s and T_c can be calculated as

$$T_s = T_{RTS} + T_{CTS} + T_{DATA} + T_{ACK} + 3 \times T_{SIFS} + T_{DIFS} \tag{3.11}$$

$$T_c = T_{RTS} + T_{EIFS} \tag{3.12}$$

Let Z be the data frame transmission time, similar to X, we can have

$$\begin{aligned} E[Z] &= \sum_{n=0}^{m-1} (1-P_c){P_c}^n n T_c + (1-{P_c}^m)T_s + {P_c}^m T_c \\ &= \frac{(m-2){P_c}^{m+1} - (m-1){P_c}^m + P_c}{1-P_c} \cdot T_c + (1-{P_c}^m)T_s \end{aligned} \tag{3.13}$$

Then the overall service time $1/\mu$ can be calculated as

$$E[1/\mu] = E[B] + E[Z] \tag{3.14}$$

Note that $E[1/\mu]$ is the mean of the service time. In other words, it is average time from when a packet is at the head of the MAC queue to the moment it is received. In order to obtain the overall transmission time L, which is also the instant reward, we still need to consider the waiting time in the queue:

$$E[L] = E[1/\mu] + \overline{W}(\lambda, \mu) \tag{3.15}$$

where $\overline{W}(\lambda, \mu)$ is the mean waiting time. If we simplify the queueing model as a M/M/1/K queue, then

$$\overline{W}(\lambda, \mu) = \begin{cases} \frac{1/\mu}{1-\lambda/\mu} - \frac{(\lambda/\mu)^{K+1}}{1-(\lambda/\mu)^{K+1}} \cdot \frac{K+1}{\lambda} & \lambda \neq \mu \\ \frac{K}{2\lambda} & \lambda = \mu \end{cases} \tag{3.16}$$

where λ is the packet arrival rate and K is the maximal length of the queue.

It can be seen that the total delay L can be work out once the number of neighbors v (μ is a function of v) and the packet arrival rate λ are known.

3.1.3 Modified Model

The assumption of homogeneous data rate and full load made by the CSMA/CA model above is not realistic in MANETs. Moreover, in order to count the number of the fully loaded neighbors, a predefined threshold, which can only be determined heuristically, is used to differentiate saturated and non-saturated nodes. As a result, the non-saturated delay estimation is no longer accurate. Therefore, in this section, we redefine the CSMA/CA model which allows arbitrary traffic loads for each node.

Let $T = [\tau_1, \tau_2, \tau_3, ...\tau_n]$ be the average attempts per slot for all the nodes. We assume that each node is aware of its own τ value by observing the MAC layer transmission queue. Since transmission attempts of node i are affected by its interfering neighbors $\overleftarrow{Adj}_i$, the first step is to define the joint probability of concurrent transmission attempts $\kappa^{(n)}_{\overleftarrow{Adj}_i}$, which represents the probability of n spontaneous transmission attempts among the node set $\overleftarrow{Adj}_i$ at any time slot. The special case when $n = 0$ and $n = 1$ can be calculated as

$$\kappa^{(0)}_{\overleftarrow{Adj}_i} = \prod_{j \in \overleftarrow{Adj}_i} 1 - \tau_j \tag{3.17}$$

$$\kappa^{(1)}_{\overleftarrow{Adj}_i} = \sum_{j \in \overleftarrow{Adj}_i} \tau_j \prod_{k \in \overleftarrow{Adj}_i, k \neq j} 1 - \tau_k \tag{3.18}$$

It can be seen that $\prod_{j \in \overleftarrow{Adj}_i} 1 - \tau_j$ is the probability that all the neighbors of node i keep silent in a time slot. Therefore, its complement is the probability that at least one node is transmitting. The average probability of transmission attempts per time slot made by $\overleftarrow{Adj}_i$, which is denoted as $\widehat{\tau}_i$, can be calculated as

$$\widehat{\tau}_i = 1 - \kappa^{(0)}_{\overleftarrow{Adj}_i} \tag{3.19}$$

The collision probability for each transmission attempt made by node i is denoted by $P_c(i)$, which is also the probability that at least one of node i's neighbors is transmitting when node i transmits:

$$P_c(i) = \widehat{\tau}_i \tag{3.20}$$

Based on the same terminology, $\widehat{P}_c(i)$ is defined as the collision probability of each transmission attempt made by a node in set $\overleftarrow{Adj}_i$, which can be calculated as the probability that more than one node transmits in $\overleftarrow{Adj}_i$ given at least more than one node transmits in $\overleftarrow{Adj}_i$:

$$\widehat{P}_c(i) = \frac{1 - \kappa^{(0)}_{\overleftarrow{Adj}_i} - \kappa^{(1)}_{\overleftarrow{Adj}_i}}{1 - \kappa^{(0)}_{\overleftarrow{Adj}_i}} \tag{3.21}$$

Similar to (3.9), the average backoff delay X_i without the counter pause time can be calculated as

$$E[X_i] = \begin{cases} \frac{(CW_{min}+1)(2^m P_c(i)^m - 1)}{4P_c(i) - 2} & P_c(i) \neq 0.5 \\ \frac{(CW_{min}+1)m}{2} & P_c(i) = 0.5 \end{cases} \tag{3.22}$$

Then the service time $1/\mu_i$ of node i is

$$E[1/\mu_i] = E[X_i] + E[X_i]\widehat{\tau}_i(\widehat{P}_c(i)T_c + (1 - \widehat{P}_c(i))T_s) \tag{3.23}$$

By substituting the $E[X]$ with $E[X_i]$ in (3.15), we can obtain L_i, the total

delay of node i.

3.1.4 Multi-hop Extension

It can be seen that L_i is a function of the current packet arrival rate λ_i and the average transmission attempts T. Therefore, if λ_i and T remain unchanged, the delay L_i will also remain unchanged. However, since it has nothing to do with the receiver, the transmission delays to all the neighbors are the same if we adopt the CSMA/CA model above. In that case, we can simply obtain the delay feedback from one of the neighbors as in chapter 2. Then why do we need the CSMA/CA delay model in the first place?

The short answer is that the transmission delay are also affected by the receiver side, because of the hidden node problem. Figure 3.3 shows an example of the hidden node problem: although both node A and C can communicate with node B, but they are hidden from each other. As a result, the transmission between A and B fails when C is also transmitting. In fact, the RTS/CTS handshaking mechanism is designed to mitigate this problem: when receiving the NAV information, nodes within both node A's or B's transmission range will keep silent until the current transmission finishes. As a side effect, node A and C have to compete with each other although they are not neighbors.

In fact, it is quite unlikely that the sender and receiver share the same neighbors in a MANET. Therefore, in order to calculate $1/\mu_i$, we need to obtain the traffic information from neighbors of both the sender and receiver because they all participate in contention. First we need to extend the definition of neighbors. Let $\overleftarrow{Adj}_{i,l}$ be the joint set of neighbors, where node i is the source and node l is

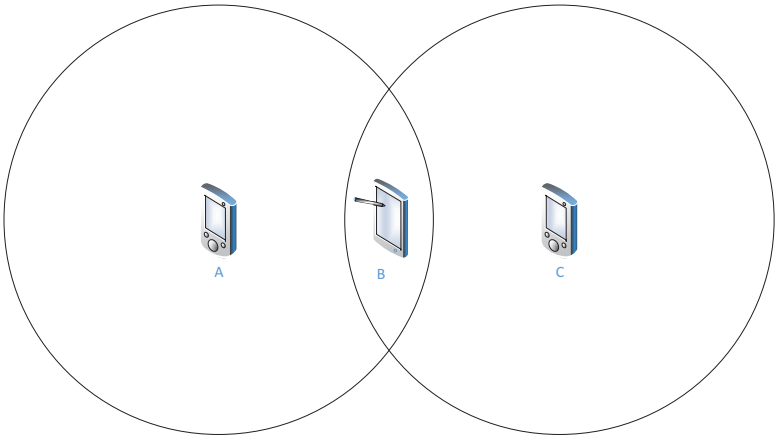


Fig. 3.3: Hidden node problem

the destination. It can be defined as

$$\overleftarrow{Adj}_{i,l} = \{k : k \in \overleftarrow{Adj}_i \cup \overleftarrow{Adj}_l, k \neq i\} \tag{3.24}$$

Note that the extra condition $k \neq i$ excludes only i but not l because when the sender tries to sending packets to destination, the destination may also has packets to send. Therefore, they still have to compete with each other. $\widehat{\tau}_{(i,l)}$ and $\widehat{P}_{c(i,l)}$ can be defined as

$$\widehat{\tau}_{(i,l)} = 1 - \kappa^{(0)}_{\overleftarrow{Adj}i,l} \tag{3.25}$$

$$\widehat{P}_{c(i,l)} = \frac{1 - \kappa^{(0)}_{\overleftarrow{Adj}_{i,l}} - \kappa^{(1)}_{\overleftarrow{Adj}_{i,l}}}{1 - \kappa^{(0)}_{\overleftarrow{Adj}_{i,l}}} \tag{3.26}$$

The mutual neighbors of both nodes i and l have to be determined to avoid

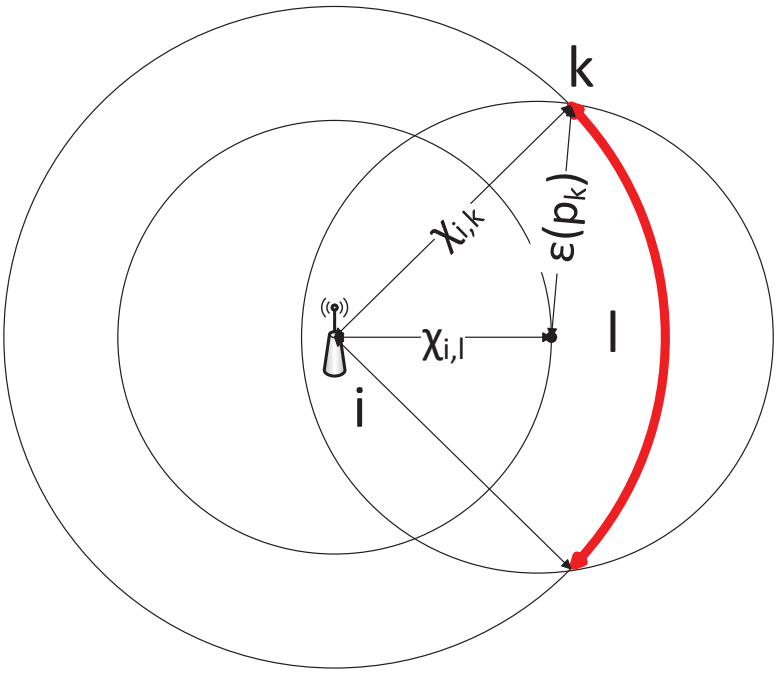


Fig. 3.4: An example of how to determine a mutual neighbor

being double counted. One simple solution is to explicitly compare $\overleftarrow{Adj}_i$ and $\overleftarrow{Adj}_l$, which inevitably introduces large amounts of extra communication overhead. Another feasible solution is to obtain a reasonable estimate based on a stochastic model.

For each interfering neighbor of node i, its probability of being a mutual neighbor can be estimated based the geometry calculation as shown in Fig. 3.4. In this example, we try to determine whether node k is a mutual friend of both node i and l. Assume that $\chi_{i,l}$ and $\chi_{i,k}$ are the distances between i, l and i, k which are assumed to be known to node i for now. Therefore, the two circles centered at node i represent all the possible locations of node l and k. Without losing generality, we can fix node l and consider only node k. On the other hand, $k \in \overleftarrow{Adj}_l$ implies that $\chi_{i,l} < \epsilon(p_k)$, where $\epsilon(p_k)$ is the transmission range of node k with the transmission power p_k. The third circle centered at node l indicates

all the possible locations of node k where node l can be reached. Clearly the red arc indicates all the possible locations of node k being a mutual neighbor. The probability can be expressed as

$$\zeta_{i,l}^{k} = \arccos(\frac{\chi_{i,k}^{2} + \chi_{i,l}^{2} - \epsilon(p_k)^{2}}{2\chi_{i,k} \times \chi_{i,l}})/(2\pi) \tag{3.27}$$

note that here $k \neq l$. If $k = l$, then $\zeta_{i,l}^{k} = 0$.

Then we can define $\overline{\kappa}_{\overleftarrow{Adj}_i}^{(0)}$ and $\overline{\kappa}_{\overleftarrow{Adj}_i}^{(1)}$, which is the estimated value of $\kappa_{\overleftarrow{Adj}_l}^{(0)}$ and $\kappa_{\overleftarrow{Adj}_l}^{(1)}$ by considering the effect of mutual neighbors, respectively:

$$\overline{\kappa}_{\overleftarrow{Adj}_i}^{(0)} = \prod_{j \in \overleftarrow{Adj}_i} 1 - \tau_j + \zeta_{i,l}^{j}\tau_j \tag{3.28}$$

$$\overline{\kappa}_{\overleftarrow{Adj}_i}^{(1)} = \sum_{j \in \overleftarrow{Adj}_i} (1 - \zeta_{i,l}^{j})\tau_j \times \prod_{k \in \overleftarrow{Adj}_i, k \neq j} 1 - \tau_k + \zeta_{i,l}^{k}\tau_k \tag{3.29}$$

Note that $i \in \overleftarrow{Adj}_l$. Since node i cannot compete with itself, we also need to exclude node i from $\overleftarrow{Adj}_l$ when calculating κ:

$$\kappa_{\overleftarrow{Adj}_{l-i}}^{(0)} = \kappa_{\overleftarrow{Adj}_l}^{(0)}/(1 - \tau_i) \tag{3.30}$$

$$\kappa_{\overleftarrow{Adj}_{l-i}}^{(1)} = (\kappa_{\overleftarrow{Adj}_l}^{(1)} - \tau_i\kappa_{\overleftarrow{Adj}_{l-i}}^{(0)})/(1 - \tau_i) \tag{3.31}$$

It is time to calculate $\kappa_{\overleftarrow{Adj}_{i,l}}^{(0)}$ and $\kappa_{\overleftarrow{Adj}_{i,l}}^{(1)}$ in (3.25):

$$\kappa_{\overleftarrow{Adj}_{i,l}}^{(0)} = \overline{\kappa}_{\overleftarrow{Adj}_i}^{(0)} \kappa_{\overleftarrow{Adj}_{l-i}}^{(0)} \tag{3.32}$$

$$\kappa_{\overleftarrow{Adj}_{i,l}}^{(1)} = \overline{\kappa}_{\overleftarrow{Adj}_i}^{(1)} \kappa_{\overleftarrow{Adj}_{l-i}}^{(0)} + \overline{\kappa}_{\overleftarrow{Adj}_i}^{(0)} \kappa_{\overleftarrow{Adj}_{l-i}}^{(1)} \tag{3.33}$$

By replacing $\kappa_{\overleftarrow{Adj}_i}^{(0)}$ and $\kappa_{\overleftarrow{Adj}_i}^{(1)}$ with $\kappa_{\overleftarrow{Adj}_{i,l}}^{(0)}$ and $\kappa_{\overleftarrow{Adj}_{i,l}}^{(1)}$ in the previous section, we can work out the specific end-to-end delay $L_{i,j}$ in exactly the same way.

The overall procedure of calculating the specific delay to all the neighbors is listed in Algorithm 4. It will be executed whenever a broadcasted Hello packet is received from the neighbors.

Algorithm 4 Procedure of updating the end-to-end delays of the neighbors

function ReceiveProbingPackets($HelloPacket$ from node j)
    Extract τ_j , $\kappa^{(0)}_{\overleftarrow{Adj_j}}$ and $\kappa^{(1)}_{\overleftarrow{Adj_j}}$ from $HelloPacket$
    Update $\kappa^{(0)}_{\overleftarrow{Adj_i}}$ and $\kappa^{(1)}_{\overleftarrow{Adj_i}}$ according to (3.17)
    for all node $l \in \overleftarrow{Adj_i}$ **do**
        Calculate $\overline{\kappa}^{(0)}_{\overleftarrow{Adj_i}}$ and $\overline{\kappa}^{(1)}_{\overleftarrow{Adj_i}}$ according to (3.28)
        Calculate $L_{i,j}$ and update
    end for

end function

3.1.5 Distance Estimation

Location information is required in (3.27) to obtain ζ. Although the location information can be obtain from the location discovery schemes [71–73], it is costly and not necessary. Instead, the relative location information, more specifically, the ratios of all the distances, is enough for us to work out ζ. We can assume that the expected signal gain r_G is exponentially proportional to the distances χ :

$$E(r_G) \propto \chi^{(-n_l)} \tag{3.34}$$

where n_l is the path loss exponent with the value between 2 to 4. n_l can be determined by empirical data or the path loss exponent estimation method [74].

To estimate $E(r_G)$, nodes have to continuously record the received and transmission power level from its neighbors and smooth them using a moving average

function to overcome the problem of varying interference, signal fading and mobility. Let $r_G^{i,j}$ denote the instantaneous reading of the signal gain from the node j. Its corresponding estimated value $\hat{r}_G^{i,j}$ can be recursively updated:

$$\hat{r}_G^{i,j} \leftarrow MA(r_G^{i,j}, \hat{r}_G^{i,j}) \tag{3.35}$$

where MA is a moving average function. A typical choice would be the Exponential Moving Average (EMA) as in [57]. This process is repeated at discrete time intervals.

By replacing the distance χ with the signal gain r_G, (3.27) can be rewritten as:

$$\zeta_{i,l}^k(p_i) = \arccos\left(\frac{(\hat{r}_G^{i,k})^{-2n_l} + (\hat{r}_G^{i,l})^{-2n_l} - (\frac{p_{ET}}{p_k})^{-2n_l}}{2(\hat{r}_G^{i,k})^{-2n_l} \times (\hat{r}_G^{i,l})^{-2n_l}}\right)/(2\pi) \tag{3.36}$$

where p_{ET} is the energy detection threshold.

3.1.6 Routing Loop Prevention with End-to-End Information

Although the routing loop prevention mechanism described in chapter 2 can still be used, it burdens the network by introducing the extra probing packets periodically. If the network is heavily congested, it is quite likely that the probing packets are dropped without even reaching its destination.

In fact, with the end-to-end delay information obtained from the learning process, there is a more efficient and reliable method that prevents loops from

forming. Two policies are adopted to prevent the routing-loop problem: spatial and temporal policies. The spatial rule indicates that for any newly added routing entry, there must be another entry that either takes more hops or takes more time to reach the destination. This is to ensure that all routing paths do not contain any node for more than once. Let us take Fig. 2.4b as an example. By using the spatial rule, Node D will not add $Q_D(B, A)$ into its routing table because $Q_D(B, A)$ has a worse Q value and a large hop count than $Q_D(B, B)$. In fact, the entry $Q_D(B, A)$ has to use node D as an intermediate node to reach destination B. That is also the reason why the routing loop is formed.

The other kind of routing loops are formed because of obsolete information. It can be addressed by the temporal policy that utilizes a sequence number to indicate the validity of neighbors information. Similar to AODV, the sequence number represents the freshness of information. For each destination, there is a corresponding sequence number maintained by the destination itself. For each fixed time interval, destination node broadcasts its information with the incremented sequence number. Any node receiving the broadcasted packets with a larger sequence number will invalidate all the corresponding entries with smaller sequence number. Then, it uses the updated sequence number to broadcast its own information.

3.1.7 Power Control

The purpose of power control is to optimize the overall network delay performance by dealing with the trade-offs between interference level and transmission range. Since multiple nodes are affected when one node changes its power level, the cost function of power control has to involve multiple Q entries of different nodes as well.

At first, Let us consider the effect of transmission range. As explained earlier, for any node i, its transmission delay is determined by its interfering neighbors $\overleftarrow{Adj}_i$, which is independent of the node i's transmission power p_i. Instead, $\overrightarrow{Adj}_i$, which indicates how many neighbors are affected by node i, is directly determined by p_i . Since $Adj_i = \overleftarrow{Adj}_i \cap \overrightarrow{Adj}_i$, we can derive that $Adj_i(p_i) \geq Adj_i(p_i')\forall p_i > p_i'$, where $Adj_i(p_i)$ is the set of the actual neighbors when the transmission power level is p_i. In other words, using a higher transmission power level results in more routing options in spite of its higher energy consumption. However, it does not mean higher power level is better. For example, if $\overleftarrow{Adj}_i \in \overrightarrow{Adj}_i$, increasing power level is purely a waste of energy and bandwidth because no benefit can be obtained from the increased transmission range.

Interference level is also observable. In fact, the set $\overrightarrow{Adj}_i$ contains all the nodes affected by the node i. The interference level of a power level can be measured by summing up all interference caused in $\overrightarrow{Adj}_i$.

Let us first define a cost function for power control:

$$C_p(p_i) = C_r(p_i) + C_i(p_i) \tag{3.37}$$

where $C_r(p_i)$ and $C_i(p_i)$ are cost functions of transmission range and interference level with transmission power level p_i, respectively.

As mentioned earlier, the interference level can measured by the weighted delay which is transmission rate times its expected delay. $C_r(p_i)$ is straight forward to calculate, which is just the sum of all the weighted delays for all the traffic flows passing through it:

$$C_r(p_i) = \sum_{d \in D} \lambda_i^d \max_{a \in Adj_i(p_i)} (Q(d, a)) \tag{3.38}$$

where λ_i^d is the packet arrival rate for node i destined to node d.

Based on the same terminology, $C_i(p_i)$ can be defined as the extra weighted delay caused to its neighbors, which can be represented in terms of C_r:

$$C_i(p_i) = \sum_{j \in \overrightarrow{Adj}_i(p_i)} \Delta C_r(p_j) \tag{3.39}$$

where $\Delta C_r(p_j)$ is the extra weighted delay caused by including node j in $\overrightarrow{Adj}_i(p_i)$. Calculating the exact value of $\Delta C_r(p_j)$is not practical nor necessary since it requires lots of information from node j. we can use a first order Taylor series approximation instead:

$$C_i(p_i) \approx \sum_{j \in \overrightarrow{Adj}_i(p_i)} \tau_i \times \frac{\partial C_r(p_j)}{\partial \tau_i} = \tau_i \sum_{j \in \overrightarrow{Adj}_i(p_i)} \frac{\partial C_r(p_j)}{\partial \tau_i} \tag{3.40}$$

Let $I_p(j) = \frac{\partial C_r(p_j)}{\partial \tau_i}$, where $I_p(j)$ is the *interference price* of node j. Therefore, once node i receives the interference prices of its neighbors, the optimal power level therefore can be obtained by:

$$p_i^* = \underset{p_{\min} < p_i < p_{\max}}{\arg\max} \; C_p(p_i) \tag{3.41}$$

The detailed calculating is explained in Algorithm 5.

The only task left is to calculate $I_p(j)$, which can be approximated as:

$$
\begin{aligned}
I_p(j) = \frac{\partial C_r(p_j)}{\partial \tau_i} &= \frac{\partial \sum_{d \in V} \lambda_j^d \max\limits_{a \in Adj_j(p_j)} (Q(d,a))}{\partial \tau_i} \\
&\approx \frac{\partial \sum_{d \in D} \lambda_j^d D_j}{\partial \tau_i} \\
&= \frac{\partial D_j \sum_{d \in D} \lambda_j^d}{\partial \tau_i} \\
&= \frac{\partial D_j \lambda_j}{\partial \tau_i}
\end{aligned} \tag{3.42}
$$

The approximation made in (3.42) assumes that the sender and receiver share the same neighbors. Therefore, the problem is reduced to calculate the derivative of D_j, which then can be further reduced by using the differentiation chain rule. In the end, we just need to calculate the derivative of $\kappa^{(0)}_{\overleftarrow{Adj_j}}$ and $\kappa^{(1)}_{\overleftarrow{Adj_j}}$.

The derivatives can be obtained by adding a *virtual* neighbor i with $\tau_i = 0$. Therefore, we can get the following equations:

$$
\frac{\partial \kappa^{(0)}_{\overleftarrow{Adj_j}}}{\partial \tau_i} = \frac{\partial [(1-\tau_i) \prod\limits_{k \in \overleftarrow{Adj_j}} 1-\tau_k]}{\partial \tau_i} = - \prod_{k \in \overleftarrow{Adj_j}} 1-\tau_k \tag{3.43}
$$

$$
\frac{\partial \kappa^{(1)}_{\overleftarrow{Adj_j}}}{\partial \tau_i} = \prod_{k \in \overleftarrow{Adj_j}} 1-\tau_k - \sum_{k \in \overleftarrow{Adj_j}} \tau_k \prod_{l \in \overleftarrow{Adj_j}, l \neq k} 1-\tau_l \tag{3.44}
$$

You may notice that (3.43) only considers the case when node i is not node j's interfering neighbor. What if $i \in \overleftarrow{adj_j}$? We choose to use (3.43) even if $i \in \overleftarrow{adj_j}$ for mainly two reasons:

- The difference between these two cases is small. For example, if $i \in \overleftarrow{adj_j}$, then $(1-\tau_i)' = -1$. If we use (3.43), $(1-\tau_i)' = -(1-\tau_i)$. Let us consider the worst case that, node i transmits with full load without any contention.

The typical value of CW_{min} is 32 in IEEE 802.11b. Therefore, we can get $\tau_i = 2/33 = 0.061$. Therefore, the error is also 6.1%. Typically, τ is much smaller than this value because of the contention from neighbors.

- By using (3.43), the final result is independent of node i. In other words, the interference price calculated, $I_p(j)$, can be used by any node. Therefore, we can just broadcast the price without specifying the receiver.

Note that it is possible to use a higher order approximation in (3.40). Besides, in (3.42), we can also get rid of the approximation by using $D_{j,k}$ instead of D_j to obtain the derivative. In that case, we need to calculate the derivative of $\kappa^{(0)}_{\overleftarrow{Adj}_{j,l}}$ and $\kappa^{(0)}_{\overleftarrow{Adj}_{j,l}}$ $\forall l \in Adj_j(p_j)$. The reason we choose the approximation here is that, from the heuristic results, the improvement is negligible in both cases.

Fig. 3.5 shows an example of the power control scheme. Assume node 1 has several neighbors under its maximal transmission range. There is only one flow from node 1 to node 6. The current power level p_1 (the shaded circle) is smaller than the power level p_1' (the blank circle). The power control cost function for both power levels can be expressed as:

$$C_p(p_1) = \lambda_1^6 \max(Q(6,2), Q(6,3)) + \tau_1 \times (I_p(2) + I_p(3)) \tag{3.45}$$

$$C_p(p_1') = \lambda_1^6 \max(Q(6,2), Q(6,3), Q(6,4)) + \tau_1 \times (I_p(2) + I_p(3) + I_p(4)) \tag{3.46}$$

It can be seen that when the transmission range gets larger, $C_r(p_1)$ may get smaller because the number of choices increases. However, $C_i(p_1)$, the interference level, gets higher because more neighbors are interfered. The power control scheme is aimed at finding the optimal point that can get the best trade-offs between both effects to achieve overall network optimization.

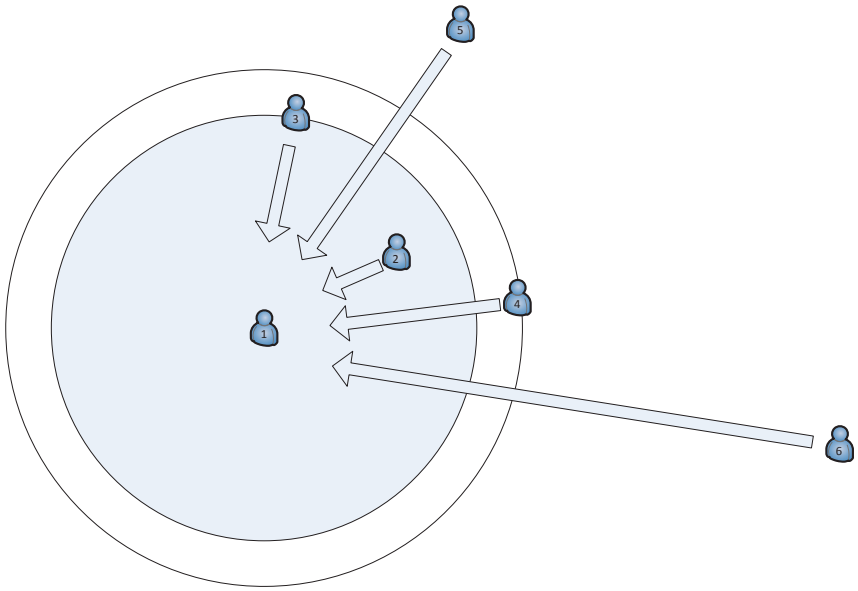


Fig. 3.5: Node 1 is trying to find the optimal power level. After sending the information requesting packets, all its neighbors sends back the response to update $\overrightarrow{Adj}_1(p_1)$). Then node 1 can choose the best power level using (3.41)

Algorithm 5 Procedure of power control

```
1: function ChooseOptimalPowerLevel
2:     BestPowerLevel = p_max
3:     C_p = inf
4:     for all p_i in available power levels do
5:         Determine Adj_i(p_i)
6:         Determine Adj_i(p_i)
7:         Calculate C_r using Adj_i(p_i)
8:         C_i = 0
9:         for all j in Adj_i(p_i) do
10:            C_i+ = Pc_j
11:        end for
12:        if C_p > C_i + C_r && BestPowerLevel > p_i then
13:            C_p = C_i + Cr
14:            BestPowerLevel = p_i
15:        end if
16:    end for
17:    Set power level to BestPowerLevel
18: end function
```

3.1.8 Information Sharing System

There are two types of information passing mechanism in our system: passive information sharing and active information requesting. General information without specific receiver is sent out using passive information sharing. To be more specific, for any node i, the following information is included in the broadcasted Hello packets using $p_{\max}$ to ensure that all its potential neighbors are notified:

- Minimal delays estimation $\max_i Q_t^a(d,i)$ from their neighbors, together with hop count information
- CSMA/CA delay model parameters, τ_i, $\kappa^{(0)}_{\overleftarrow{Adj_i}}$, and $\kappa^{(1)}_{\overleftarrow{Adj_i}}$
- Transmission power level p_i
- Interfering price $I_p(i)$

Active information requesting, on the other hand, is for the receiver-specified information. For example, in order to maintain the $\overrightarrow{Adj}_i$, each node periodically sends out information requesting packets. All the receiver respond to this request to indicate its presence. In reality, such information can also be embedded into other information requesting packets, such the RREQ packets in AODV.

3.1.9 System Analysis

The learning process is quite similar to the previous Q-learning-based routing. The phase transition diagram is shown in Fig. 3.6. Compared with the Q-learning-based routing, the major differences are:

- Learning is no longer based on individual feedback. As shown in the previous section, the convergence time is proportional to the size of the network. It is

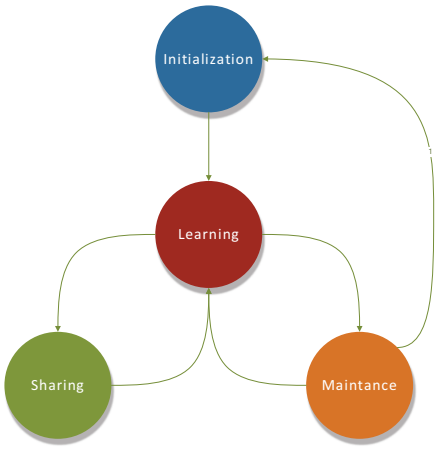


Fig. 3.6: Phase transitions diagram

expected that when the number of choice exceeds 30, it is hard to converge in a dynamic environment. Therefore, we make use of a model to obtain all the feedback at the same time when any neighbors update their status.

- Each node has to maintain another table which is used to store each neighbor's information, including CSMA/CA delay model parameters, τ_i, $\kappa^{(0)}_{\overleftarrow{Adj}_i}$, and $\kappa^{(1)}_{\overleftarrow{Adj}_i}$, Transmission power level p_i, Interference price $I_p(i)$ and required power level.

- Power control is added into the action space. However, power level is only changed at a fix time interval because frequent topology changes may result in an unstable network.

The flowchart of sending a packet is shown in Fig. 3.7. At first, nodes will try to find the corresponding Q entries. If such entries do not exist, the node will send out RREQ packets to establish an initial route. Once the destination receives the RREQ packet, it will reply with RREP packets and add itself into the HELLO message.

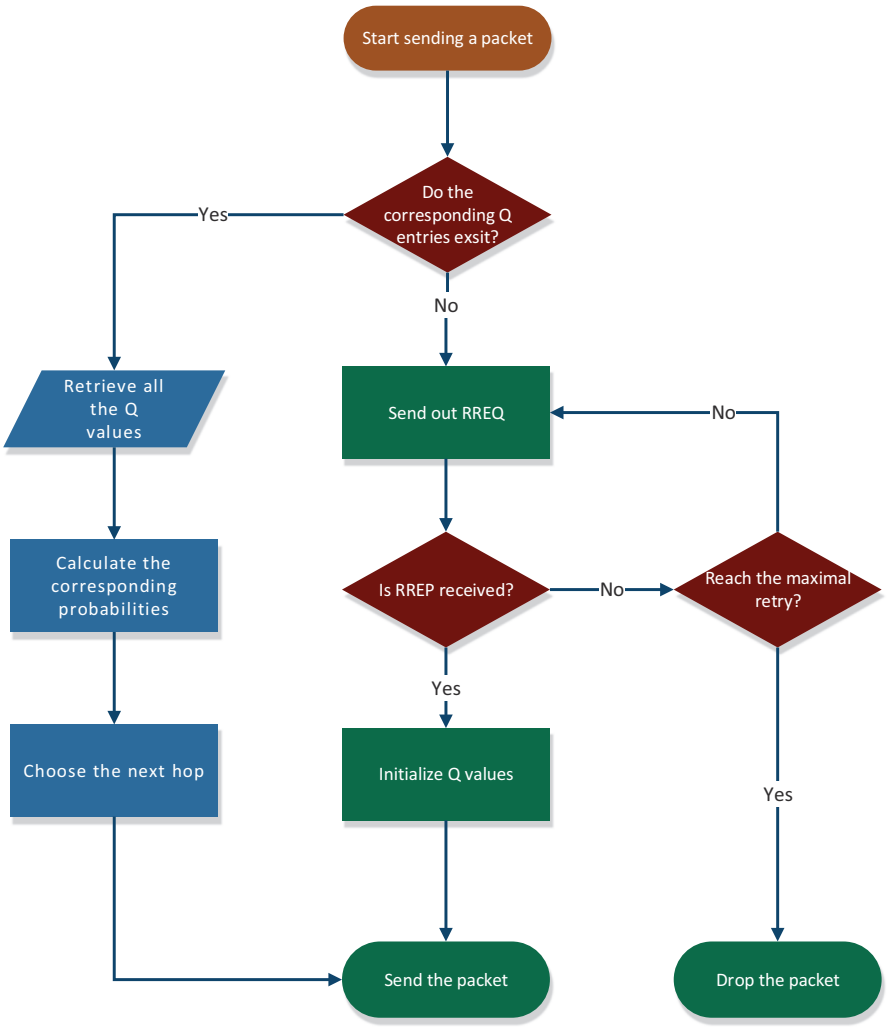


Fig. 3.7: Flowchart of sending a packet

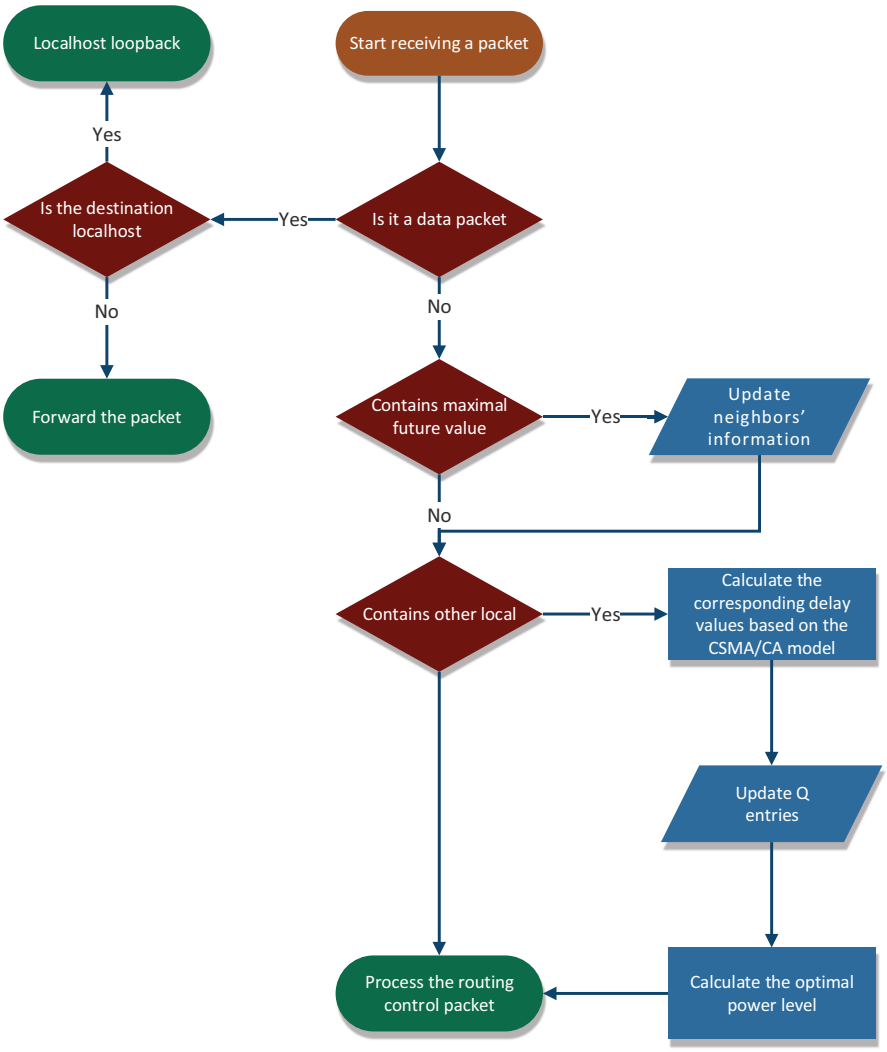


Fig. 3.8: Flowchart of receive a packet

The complete process of receiving a packet is shown in Fig. 3.8. If the received packet is a data packet, it will be forwarded or dispatched locally. If it is a routing control packet, the node has to check whether the packet contains optimal future value or the traffic information described in chapter 4.1.8 and update corresponding entries. Note that once the neighbor's information is received, all the corresponding delay is calculated to update the Q entries.

Each node needs to maintain three tables: one is the Q table which contains

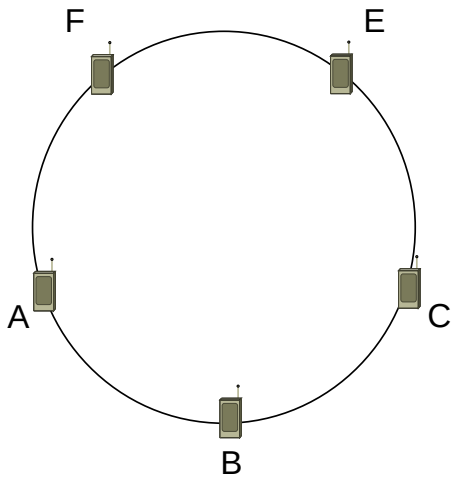


Fig. 3.9: Example of QLPCR

the Q values of all the destination-next-hop pairs. The second is the neighbor Q table which contains the optimal future values, the last one is the neighbor table containing all the neighbor information.

Entries in the neighbor Q table and the neighbor table can be added or updated only when the corresponding information is received from neighbors. These information is then used to update entries in the Q table.

As mentioned above, there are two ways to detect whether a node leaves the network. The first case is a transmission failure when the number of retry attempts reaches the optimal value. The other case is during a timeout period there is not HELLO message received from this neighbor. Once a node is labelled as absent, all the corresponding entries from the three tables are removed.

We will use the example in chapter 1 to illustrate how QLPCR works in a small scale network.

Let node A and node C be the source and destination respectively. Initially there is no route established.

When A tries to send packets to C, it will find that no Q entries are available.

Therefore, it sends out a RREQ packet to find the destination, once the RREP packet is received, it will add the new entry into the Q table:

Table 3.1: Q Table of Node A

Destination	Next hop	Q value
C	B	1

The initial Q value is set to 1 to encourage exploration of the new path. Although node F also sends RREP back to node A, since it requires one more hop to reach C, the route is discarded in the initial route establishing process.

Once traffic starts to generate in the network, each node will monitor its traffic conditions and broadcast the information. Meanwhile optimal future values are also shared among neighbors through HELLO message. For example, after some time, node A will receive its neighbor's maximum future values and store them in the neighbor Q table:

Table 3.2: Neighbor Q Table of Node A

Destination	Next hop	Maximum future value
C	F	0.726
C	B	0.635
C	C	1
...		

Note that the neighbor Q table is updated only when newer HELLO message is received from the neighbors.

Correspondingly, the Q is also updated by observing the traffic information and the neighbor Q value table. The delay value, in this case, also the reward R_a can be calculated by the equation (4.13), then the value of R_a is used in (3.5) to obtain the new Q value. Suppose after some time, the Q table of node A is Note that although $Q_A(C, C)$ has the smallest value here, node C is not reachable

Table 3.3: Q Table of Node A After Update

Destination	Next hop	Q value
C	F	0.578
C	B	0.336
C	C	0.898
...		

for the current power level. Therefore this entry is excluded when calculating the corresponding routing probability by equation (3.7). In this case, we can get $Pr(R_A(C) = F) = 0.56$.

If node F leaves the network, all the entries containing F are removed. As a result, only node B is chosen as the next hop.

Power control is performed periodically, usually every 20 seconds or longer. By using Equation (4.34), we can obtain the cost function for different power levels. In this case, if the interference prices of node F and C are small, node A will increase its power level to reach node C directly. Whereas if F and C are heavily loaded, it is quite unlikely for A to increase its power level because of the higher interference prices.

The most significant strength for QLPCR is its capability to adjust routing strategy and power level based on surrounding traffic conditions to maximize the overall network performances.

Its weakness is the operational cost which involves extra storage, computation and information sharing. In a sparse or lightly loaded network, it may not be worth the effort for such an optimization. More importantly, since all the destinations are maintained for each node, the overhead complexity increases proportional to the size of a network.

Another weakness is that the operation of QLPCR highly depends on the shared information. Therefore once the packet delivery ratio gets low, it is hard

to recover from such a scenario.

3.2 Performance Evaluation

In this section, we evaluate the QLPCR protocol with the same setting as in chapter 2. Three benchmarks are chosen from three categories: LMN [23] from the power control schemes, QLMAODV [46] from the Q-routing schemes and PCR [53] from the power-controlled routing schemes.

LMN actively changes the transmission power so that each node has to maintain its number of neighbors as the mean value of its neighbors. PCR is a combination of power control and routing schemes. In order to choose the best power level, its routing table is constructed and maintained for each power level. Therefore, nodes have to constantly change their power levels for exploration to update the routing entries. It uses the link weight which is defined as the number of neighbors, as the routing metrics to minimize the interference level.

In addition to the existing metrics, we also measure the power consumption rate because power control inevitably has great impacts on the power consumption rate. The detailed power consumption data is shown in Table 3.4 [57]:

Table 3.4: Detailed Power Consumption

Idle power consumption	0.6699 W
Reception power consumption	1.049 W
Output power levels	[0.01, 0.013, 0.02, 0.025, 0.04] W
Power consumption for electronics	1.6787 W
Energy detection threshold	-96 dbm

For each packet transmission, we assume that the final power consumption is the sum of output power levels and the constant power consumption for electronics

[75]. The total power consumption over a certain time period is mainly based on the duration of each state, namely idle, receiving and transmitting. Therefore, it is possible that a higher output power level may result in lower power consumption if the transmitting period can be shortened.

Similar to the previous chapter, we analyze all the four protocols under different node densities, traffic loads, and mobility levels. The settings are almost identical to chapter 2 except that the data rate is reduced to 8 packets/s because some measurements are too low if 10 packets/s is used.

3.2.1 Node Density

In this scenario, the number of nodes is increased from 50 to 100. It is expected that higher node density will impair the overall network performances because of the increased communication overhead and contention. The simulation results are shown in Fig. 3.10.

Fig. 3.10a shows that PCR has the longest delay because of its minimum link weight algorithm: the nodes with different traffic load are treated equally. As a result, the nodes in the spare areas are preferred even if they are overloaded. LMN outperforms QLMAODV because of the simple effective interference control scheme with minimum communication overhead. QLPCR has a similar performance to LMN, thanks to its power control scheme that minimizes the overall interference level.

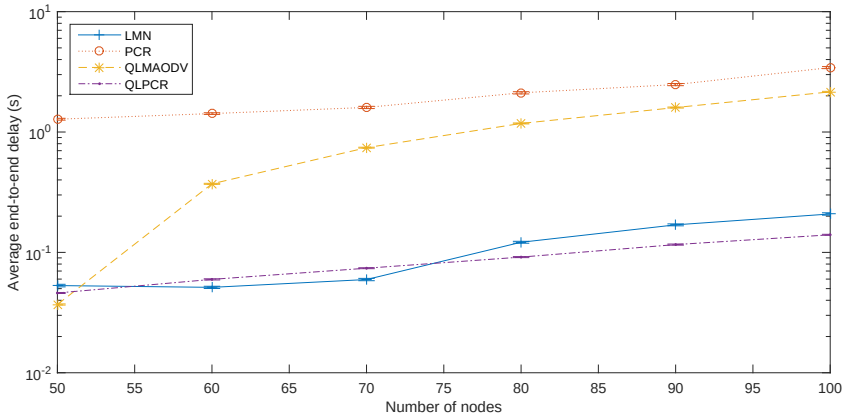
PDR drops when number of nodes increases as shown in Fig. 3.10b. It can be seen that QLMAODV has a significant drop in PDR when the number of nodes reaches 70, due to the lack of a power control scheme. As a result, the nodes in dense areas suffer tremendous interference. All the power-controlled schemes, on the other hand, although still suffering from interference, limits its damage to

an acceptable level. QLPCR jointly considers both routing and power control, by takes the real-time traffic conditions into account when deciding the optimal power level. Therefore, as long as the traffic load does not increases, its performance will not be greatly affected by the node density. Although PCR has the lowest PDR value, it can be seen that its power control scheme is still capable of controlling the interference level when the node density gets higher. LMN has a fair performance in PDR by limiting the number of neighbors.

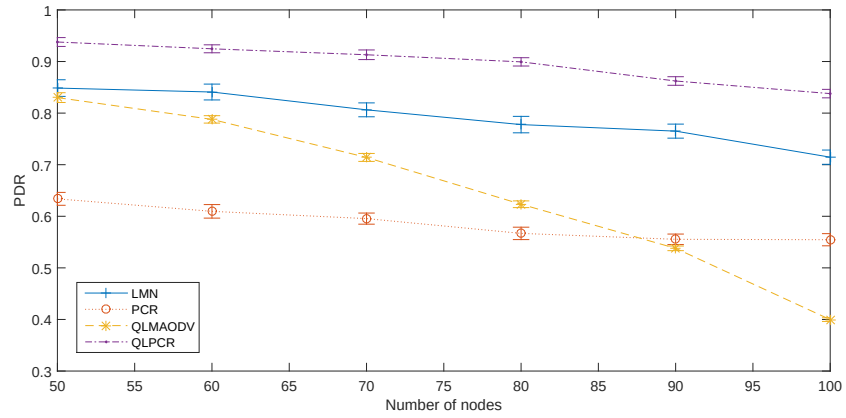
QLPCR and PCR have a relatively stable energy consumption performance because their power levels are not determined by the node density (Fig. 3.10c). LMN, on the other hand, has to decrease the power level in a high density environment to maintain the number of neighbors. That is also the reason why its energy consumption drops when the node density increases. Although QLMAODV has the highest power consumption rate initially, it drops dramatically while the number of nodes increases. This is mainly due to its low PDR value so that only a smaller portion of time is used for the data transmission.

3.2.2 Traffic Load

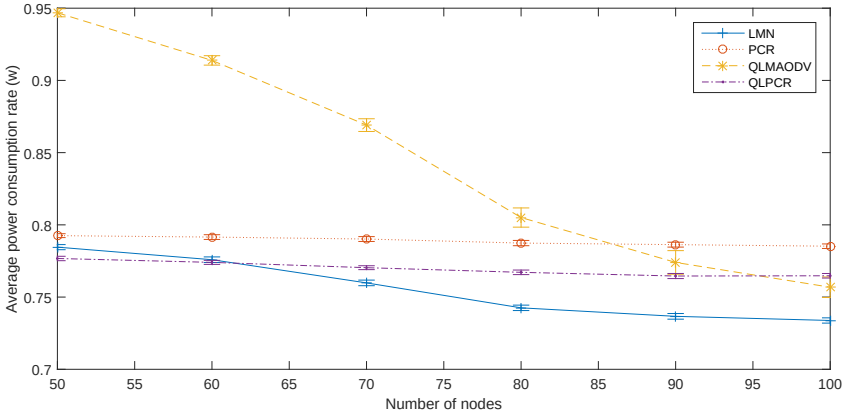
In this scenario, the number of nodes is fixed at 50 while the number of flows increases from 5 to 40. All the protocols have a significant PDR drop because of the network congestion (Fig. 3.11a). Among them, QLMAODV has the largest drop of around 50%. This is a good example to show why power control is essential in the QoS optimization: although the Q-learning-based routing method is capable of avoiding congestion areas, when the whole network gets congested due to the excessive interference, there is not so much a routing scheme can help. QLPCR has the best performance because when the traffic rate gets higher, the interference prices get more expensive. Therefore, its transmission range is limited to reduce



(a) Delay



(b) PDR



(c) Energy consumption

Fig. 3.10: QoS performance of power-controlled schemes under different node densities

the interference cost. LMN has a similar mechanism to reduce the interference. However, it does not consider any traffic conditions. Therefore, when the traffic load gets higher, it cannot further reduce the interference.

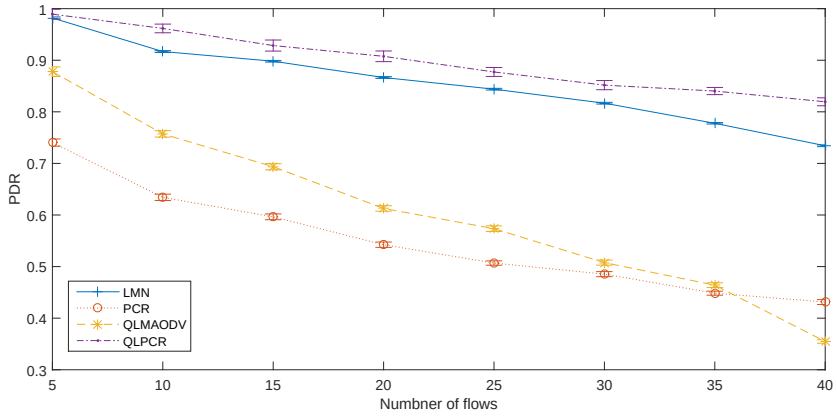
Fig. 3.11b shows the average end-to-end delay for all the protocols. It can be seen that LMN, QLPCR and QLMAODV have similar performances in this scenario. PCR has encounter a large delay when the traffic load is light because the routing decisions are heavily biased by the network topology. When the traffic load get higher, the number of neighbors has a stronger positive correlation with the interference level. This is also the reason its performance in heavy load is comparable to the one in light load.

The general trend of power consumption increases with the number of flows because there are more packets to transmit (Fig. 3.11c). As shown in Fig. 3.11a, when number of flows increases, the PDR drops accordingly. This also includes information sharing packets, which LMN makes use of to determine the number of neighbors. As a result, LMN tries to increase its power level to maintain connectivity. This is a reason why LMN has a steep slope in terms of the power consumption.

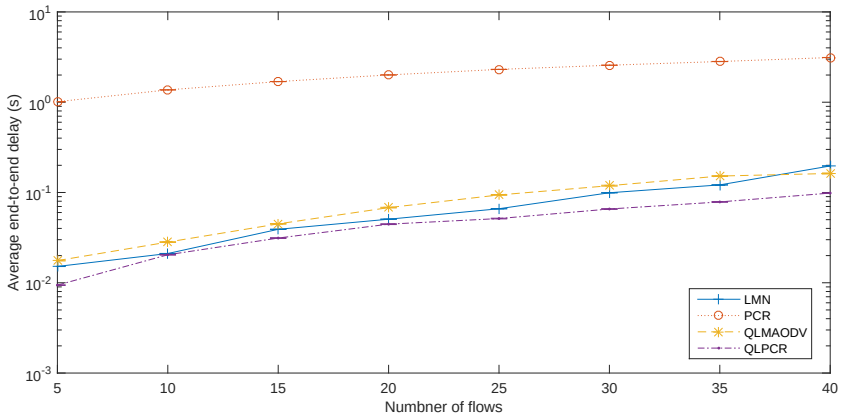
3.2.3 Mobility

Mobility indicates how fast nodes can move around in a given area with a specific mobility model. The purpose of this scenario is to evaluate the real-life situations where devices are carried by human beings. Therefore, we fix the number of nodes to 50 and number of flows to 40 and vary the maximum speed of the random way-point model from 2 m/s to 10 m/s, which covers almost the whole range of the human walking speed.

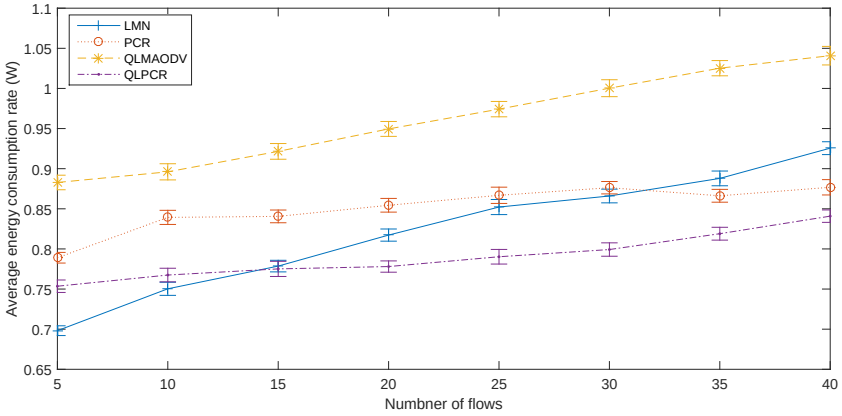
The result shown in Fig. 3.12 suggests that mobility has small impacts on



(a) PDR



(b) Delay



(c) Energy consumption

Fig. 3.11: QoS performance of power-controlled schemes under different traffic loads

LMN and PCR because they actively control the connectivity of the network to accommodate the topology change, therefore have relatively unaffected performances.

QLMAODV, on the other hand, has a significant performance drop across all the metrics. QLMAODV explicitly measures the link stability by Link Expiration Time (LET). As a result, the routing choices are heavily biased by the instantaneous moving patterns of neighbors. As a result, nodes with good link qualities are not chosen as the next hop.

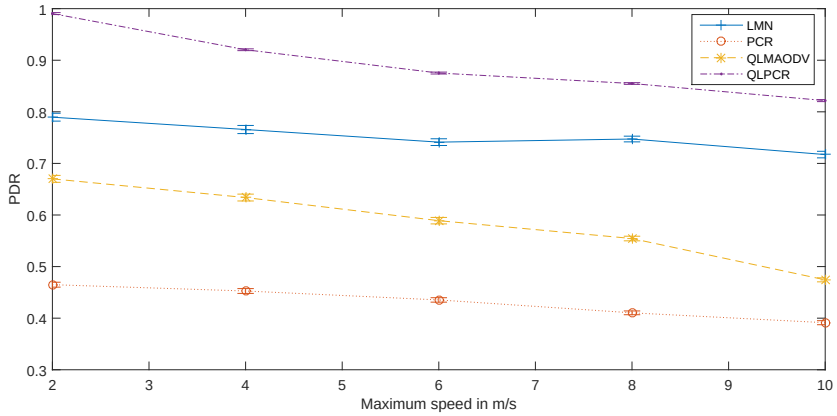
The PDR and delay performances of QLPCR also drop a lot when the node mobility increases. This is mainly due the overhead caused by the topology change. QLPCR has to actively maintain the list of neighbors by both passive information sharing and active information requesting. When the topology changes, nodes have to restart the learning process by re-sending all the information sharing packets.

3.3 Conclusion

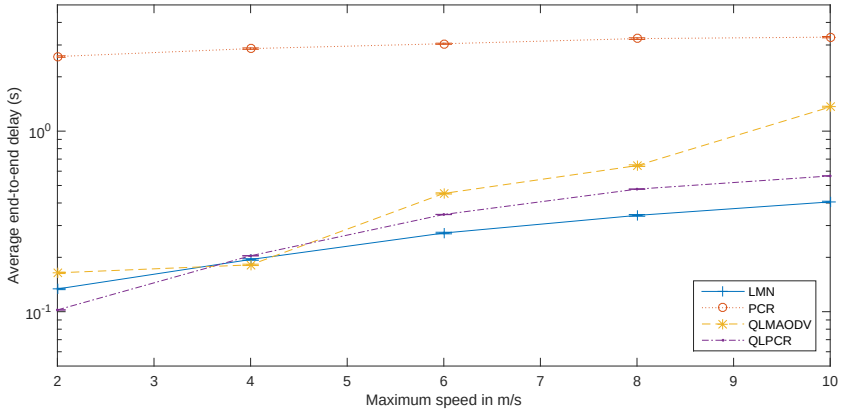
In this chapter, we manage to add power control into the Q-learning-based framework. In order to deal with the large action space formed by the power control and routing, a CSMA/CA model is introduced to simulate the learning process.

Since the instant reward is reduced to represent the delay value only, the routing-loop prevention mechanism is also improved by introducing one spatial and one chronicle policies.

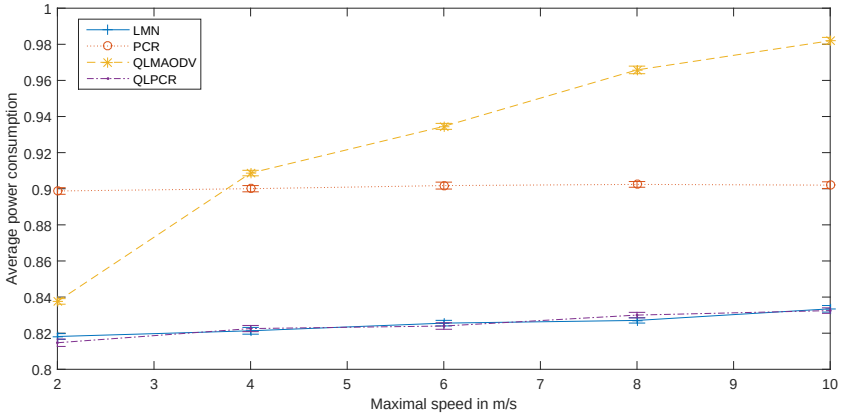
The power control scheme involves multiple nodes in a network. Therefore, we measure the interference level by a weighted delay which takes all the neighbors' traffic conditions into account. Then, we can directly compare the interference



(a) PDR



(b) Delay



(c) Energy consumption

Fig. 3.12: QoS performance of power-controlled schemes under different mobilities

level and potential performance improvement of a power level.

The simulation results show that our proposed protocol can perform well in a heavily loaded dense network with high node density. The energy consumption level is also comparable, even better at selected scenarios. However, we can still expect that it cannot perform well if the network is highly dynamic.

Chapter 4

Rate-aware Power-controlled Routing Protocol

Now we have shown how to add power control and routing into the Q-learning-based framework, leaving only one last factor: the rate adaptation. In this chapter, we explore possible solutions to combine all the three factors.

In the following content, first, we present the Q-Learning-based Rate-aware Power-Controlled Routing (QLRPCR) protocol, which extends QLPCR by adding the rate awareness. Then, we propose another solution, namely the Multi-agent Rate-aware Power-Controlled Routing (MRPCR) protocol, which constructs a decision-making system for power control, rate adaptation and routing, based on diffusion model and Distributed Constraint OPtimization (DCOP) framework. Lastly, we validate the design through simulations. The results show that MRPCR has a good QoS performance in various scenarios with reasonable power consumptions.

4.1 Protocol Design

4.1.1 Simple Rate-aware Extension in CSMA/CA Model

One easy solution to transform QLPCR into a rate-aware protocol: T_{DATA} in (3.11) can be transformed into a function of the data rate U_i:

$$T_{DATA} = \frac{L_p}{E[U_i]} \tag{4.1}$$

where L_p is the packet size.

Recall that a successful transmission must fulfill two prerequisites: (1.1) and (1.2). Therefore, the optimal transmission rate from node i to one of its neighbor j can be expressed as:

$$U_{i,j} = \begin{cases} \max\{U : T^U_{SINR} \leq \frac{p_i r_G^{(i,j)}}{p_N + p_I}\} & if\ p_i r_G^{(i,j)} \geq p_{ET} \\ 0 & if\ p_i r_G^{(i,j)} < p_{ET} \end{cases} \tag{4.2}$$

In order estimate the optimal data rate $U_{i,j}$, the receiver j needs to keep track of both p_I and r_G. We have already shown how to maintain $r_G^{i,j} \forall i \in Adj_j$ in (3.35). The same method can be applied to p_I as well. Besides, p_i is broadcasted through the Hello message, therefore node j actually has all the information required to calculate $U_{i,j}$ by (4.2), which is embedded into the ACK packet so that whenever there is a transmission between node i and j, $U_{i,j}$ can be updated.

In order to obtain the average transmission data rate for node i, we have to take all the neighbors into account and calculate the average.

Let $\lambda_i^{(d)}$ denotes the traffic rate of node i destined to node d such that $\lambda_i = \sum_{d \in D} \lambda_i^{(d)}$. Therefore, the probability of any packet destined to node d is given by $\frac{\lambda_i^{(d)}}{\lambda_i}$. Since for a given destination d, the probability of choosing node j as the next

hop is $\Pr(R_i(d) = j)$, we can further calculate the probability of node j is chosen as the next hop for any destination:

$$\Pr(R_i = j) = \frac{\sum_{d \in D} \Pr(R_i(d) = j)\lambda_i^{(d)}}{\lambda_i} \tag{4.3}$$

For a given next hop j, the traffic go through it is $\lambda_i * \Pr(R_i = j)$. Therefore, the time needed is $\frac{\Pr(R_i=j)}{U_{i,j}}$. The average data rate for node i can be calculated as

$$U_i = \frac{\lambda_i}{\sum_{j \in Adj(p_i)} \frac{\Pr(R_i=j)\lambda_i}{U_{i,j}}} = \frac{1}{\sum_{j \in Adj(p_i)} \frac{\Pr(R_i=j)}{U_{i,j}}} \tag{4.4}$$

To obtain $\Pr(Rt_i = j)$, we need to consider all the destinations and take the average:

$$\Pr(Rt_i = j) = \frac{\sum_{d \in D} \Pr(Rt_i(d) = j)\lambda_i^{(d)}}{\lambda_i} \tag{4.5}$$

where $\lambda_i^{(d)}$ denotes the traffic rate of node i destined to node d such that $\lambda_i = \sum_{d \in D} \lambda_i^{(d)}$.

Since all the Q values are updated through the model-base simulations, no exploration is required for the route discovery. Therefore, we can use the greedy approach in selecting the next hop:

$$\Pr(R_i(d) = j) = \begin{cases} 1 & if\ j = \arg\max_{k \in Adj_i} Q_i(d,k) \\ 0 & if\ j \neq \arg\max_{k \in Adj_i} Q_i(d,k) \end{cases} \tag{4.6}$$

This is not the end of the story because there are lots of room for improvement. First, interference is not caused by an excessive transmission range alone. In fact,

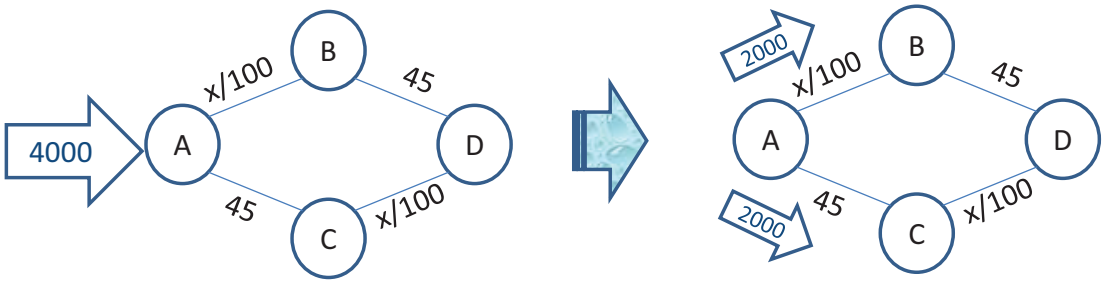


Fig. 4.1: Braess's Paradox: originally 4000 drivers need to travel from station A to D, the optimal solution is to equally divide the entire traffic

any packet transmission activity in the MANETs causes certain interference to neighbors. Therefore, considering the interference level in power control loses some potential optimization opportunities. Second, the average transmission attempts per time slot τ is not an independent random variable, although in a large network with evenly distributed traffic, it can be approximated as such. Third, multi-agent learning requires special coordination to ensure its performance.

4.1.2 Braess's Paradox

The routing strategies without considering interference leads to selfish behavior because each node only considers its own routing metrics when making the routing decisions. The interference caused in routing can be well explained by Braess's paradox, which shows how in a multi-agent environment, selfish behavior can harm the overall performance. Let us begin with the example in [76], which is a good starting point to explain Braess's paradox. Suppose 4000 drivers are going from station A to D through station B or C, the corresponding weights (in this case, time consumed) are labeled as in Figure 4.1 where x is the number of drivers going through a certain link. It is obvious that the optimal solution is to equally divide the traffic to both B and C, with the resulting average cost of 65.

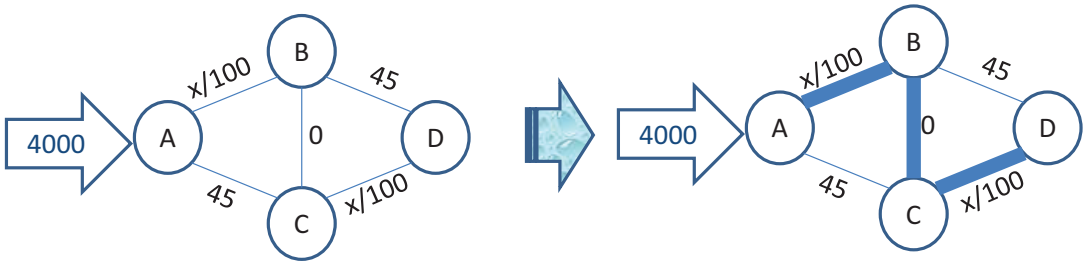


Fig. 4.2: When one extra link is added between node B and C, the final average cost increases

The situation becomes tricky when one extra path is constructed between stations B and C as in Figure 4.2. It is surprising to see that the optimal solution becomes $A-B-C-D$ if all agents just try to minimize their own costs (they are reluctant to shift from $A-B$ to $A-C$ because their local cost increases from 40 to 45). The average cost in this case is 80, which is larger than the previous case. In other words, the average performance in equilibrium is worsened by one extra path.

Braess's paradox illustrates the fact that the Nash equilibrium of a traffic network is not necessarily optimal because of interference. In this example, although each driver chooses the path with the shortest delay to the destination, form a global view, the path chosen is not optimal because the extra delay caused to existing drivers is not taken into account. In order to avoid the sub-optimal behaviors, one well-known solution is called marginal cost pricing [77]. The idea is to add an extra price into the cost function to indicate the interference level caused to others. For example, as indicated in Figure 4.3, if one driver chooses to take path $A-B$, in addition to the original cost of 40, there is an extra interference cost of 40 because the cost of another 4000 drivers are increased by 0.01. On the contrary, taking link $A-C$ costs only 45 because it does not affect other users.

Back to our problem, the same logic applies. The cost function used in our system consists of two parts: one is link cost for using a specific link and the other

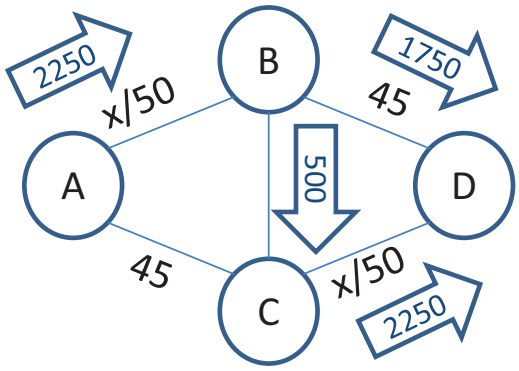


Fig. 4.3: After adjusting weight according to the global objective function, traffic will be re-distributed.

cost is interference cost representing the extra cost introduced to its neighbors. Therefore, the reward function $R_a(i,k)$ of node i can be expressed as:

$$R_a(i,k) = L_{i,k} + \sum_{j \in \overrightarrow{Adj}_i} \frac{\partial L_j}{\partial \lambda_i} \times \lambda_i \tag{4.7}$$

where $L_{i,k}$ is the actual delay obtained from chapter 3. L_j is the estimated delay going through node j. The second part is the interference level caused to node is neighbors given its transmission rate λ_i. Note that the estimated delay L_j is used to calculating the derivative only.

The marginal cost pricing mechanism moves the interference cost from power control to routing. As a result, the routing strategies now can make decisions based on the global utility. Since the Q values are also used for power control, the ability of limiting interference in power control remains unchanged.

4.1.3 Diffusion Model

Similar to the chapter 3, we need to find the derivative of the transmission delay L_i. However, the previous CSMA/CA model is based on τ, the average

transmission attempts per time slot. Since they are dependent random variables, the derivative obtained is not accurate. In this section, we propose a network model which is based on an independent random variable, the packet arrival rate λ.

The diffusion approximation is proposed as a technique to solve non-product form queuing networks [78]. By replacing the discrete valued queuing process with a continuous Markov process, which is called the diffusion process, the overall probability distribution of the state of a network can be represented by the product of the states of individual states of the individual queues. Details about queuing networks and the diffusion model can be found in [79]. The final closed-form result is simply:

$$L_i = \frac{\rho_i}{\lambda_i(1-\widehat{\rho}_i)} = \frac{\rho_i}{\lambda_i(1-e^{-\frac{2(1-\rho_i)}{{c_{Ai}}^2\rho_i+{c_{Bi}}^2}})} \tag{4.8}$$

where ρ_i is the utilization factor: the ratio between the packet arrival rate λ_i and the service rate μ_i. $\widehat{\rho}_i$ is an intermediate variable indicating the adjusted utilization factor. Note that in steady state, the packet arrival rate is the same as packet transmission rate. c_{Ai} and c_{Bi} are the coefficient of variance of the inter-arrival and service time at station i. The next step is to find $\frac{\partial L_j}{\partial \lambda_i}$ in (4.7).

In order to simplify the calculation, we assume that the random backoff timer is exponentially distributed with mean $1/\xi$ [80]. Therefore, if one station transmits without any contention, the service time $1/\mu$ would be $1/\xi + L_p/U$, where U is the average physical data rate. For simplicity, we assume that the packet size is fixed. For a node j, assume one of its neighbors, say node i, has a packet arrival rate λ_i packets / sec with constant packet length L_p bytes/packet and average physical data transmission rate U_i bits / sec. Therefore, the neighbors will make use of the

channel for transmission for $\sum_{i \in \overleftarrow{Adj}_j} \lambda_i L_p / U_i$ [80]. For node j, its packet processing time becomes:

$$\frac{1}{u_j} = \frac{1/\xi + L_p/U_j}{1 - \sum_{i \in \overleftarrow{Adj}_j} \lambda_i L_p / U_i} \tag{4.9}$$

Then if we take the derivative, the following equation can be derived:

$$\frac{\partial \rho_j}{\partial \lambda_i} = \frac{\partial (\lambda_j / u_j)}{\partial \lambda_i} = \frac{\rho_j L_p}{u_j (\frac{U_i}{\xi} + L_p \frac{U_i}{U_j})} \tag{4.10}$$

Taking the derivative of (4.8), we can obtain

$$\frac{\partial L_j}{\partial \rho_j} = \frac{(1 - \widehat{\rho}_j) + \rho_j \frac{\partial \widehat{\rho}_j}{\partial \rho_j}}{\lambda_j (1 - \widehat{\rho}_j)^2} \tag{4.11}$$

Then the derivative of D_j in terms of the data rate λ_i can be expressed as:

$$\frac{\partial L_j}{\partial \lambda_i} = \frac{\partial L_j}{\partial \rho_j} \times \frac{\partial \rho_j}{\partial \lambda_i} = \frac{(1 - \widehat{\rho}_j) + \rho_j \frac{\partial \widehat{\rho}_j}{\partial \rho_j}}{\lambda_j (1 - \widehat{\rho}_j)^2} \times \frac{\rho_j L_p}{u_j (\frac{U_i}{\xi} + L_p \frac{U_i}{U_j})} \tag{4.12}$$

since $\widehat{\rho}_j = e^{-\frac{2(1-\rho_j)}{{c_{Aj}}^2 \rho_j + {c_{Bj}}^2}}$, we can derive

$$\frac{\partial \widehat{\rho}_j}{\partial \rho_j} = e^{\frac{-2(1-\rho_j)}{{c_{Aj}}^2 \rho_j + {c_{Bj}}^2}} \cdot [\frac{2}{{c_{Aj}}^2 \rho_j + {c_{Bj}}^2} + \frac{2{c_{Aj}}^2 (1 - \rho_j)}{({c_{Aj}}^2 \rho_j + {c_{Bj}}^2)^2}] \tag{4.13}$$

In order to obtain $\frac{\partial L_j}{\partial \lambda_i}$ in (4.12), $\lambda_j, \mu_j, c_{Aj}, c_{Bj}, U_j$ are broadcasted by Hello packets using the same maximal power level so that all nodes in its maximal coverage are informed. $\lambda_j, \mu_j, c_{Aj}, c_{Bj}$ can be obtained by observing the MAC

queue. U_j, on the other hand, is determined by the rate adaptation mechanism, mentioned above.

4.1.4 Power Control Scheme

Since the interference level is embedded in the Q values, the cost function for power control can be represented as the weighted sum of all the Q values:

$$C_p(p_i) = C_r(p_i) = \sum_{d \in D} \lambda_i^{(d)} \max_{j \in Adj_i(p_i)} Q_i(d, j) \tag{4.14}$$

Note that $Adj_i(p_i)$ and $Q_i(d, j)$ are both functions of the transmission power p_i. When p_i increases, $Adj(p_i)$ contains more nodes because of larger transmission ranges. According to (4.8) and (4.9), larger power also results in lower service time and overall transmission delay because of higher transmission rate. On the other hand, it also introduces larger interference in (4.7).

4.1.5 DCOP and Multi-agent Coordination

DCOP techniques can solve the multi-agent coordination problem commonly existing in Q-learning-based methods. Generally speaking, they can be divided into complete and incomplete algorithms [81]. The main difference between these two groups is that complete algorithms can always find a global optimal solution [82–86]. However, the guarantee of optimality comes with the price of exponentially increasing coordination overhead, which severely limits their practicality [87]. On the other hand, incomplete algorithms such as the Distributed Stochastic Algorithm (DSA) [88] and Distributed Breakout Algorithm (DBA) [89], do not guarantee finding the optimal solution. Nevertheless, compared with complete algorithms, they do not require any global control mechanism. Agents in incom-

plete algorithms only require local information therefore have low computational and communication cost. More importantly, incomplete algorithms are able to make progressively improving any-time solutions by aggregating current states of individual agents. Although the optimality of solutions is not guaranteed, it is usually sufficient to finding satisfying solutions in highly dynamic environments.

The MANET is a typical multi-agent system with each node responsible of its own actions. Nodes compete for network resources and cooperate to complete packet forwarding at the same time. Without global coordination, they can only react based on local information. Unfortunately, there is no guarantee about the validity of such information if all nodes keep changing their status. That is also the reason why we need a multi-agent coordination mechanism to improve efficiency.

DSA and DBA are both incomplete algorithms requiring only local information, but at the same time, providing any-time solutions. DSA is uniform in that all agents are equal and have no identities to distinguish one another. The basic idea of DSA is straightforward. Initially all the agents pick random initial values for their variables. After that, they exchange their current state information with their neighboring agents. If their current performances can be improved by a new set of values, they will then decide stochastically to do so. The detailed algorithm is shown in Algorithm. 6. The motivation behind this is quite similar to the backoff mechanism in CSMA/CA. The backoff mechanism reduces the chance of more than one node accessing the medium. Similarly, DSA can reduce the number of nodes that change their current variables instantaneously. However, when the number of nodes is low, it may result in a slow convergence rate. For example, when only one node is capable of making improvement, DSA still requires it to act stochastically.

DBA on the other hand, is not a stochastic process. The procedure are

Algorithm 6 DSA

```
1: function DSA
2:     Randomly choose a value
3:     while Not finished do
4:         if the value is changed then
5:             Broadcast the new value
6:         end if
7:         Update the values of the neighbors
8:         if There is improvement then
9:             Select the new value stochastically
10:        end if
11:    end while
12: end function
```

shown in Algorithm 7. After receiving the neighbor's state information, each node calculates its potential improvement and broadcast it to the neighbors. The node with the greatest improvement can change its value while the rest of the nodes will just remain unchanged. By doing this, the global performance improvement can be guaranteed. However, it requires two rounds of communication and only one node within a given communications range can make the adjustment. More importantly, this algorithm can easily be trapped in a local optimal point because of the greedy approach. As a result, its performance heavily relies on the initial values.

Algorithm 7 DBA

```
1: function DBA
2:     Randomly choose a value
3:     while Not finished do
4:         Exchange value with neighbors
5:         Calculate the maximum improvement
6:         Broadcast the maximum improvement
7:         if it has the biggest improvement then
8:             change to the new value
9:         end if
10:    end while
11: end function
```

Our mechanism, Distributed Stochastic Breakout Algorithm (DSBA), combines both features of DSA and DBA. A Node with larger performance improvement has a better chance of making the change. The best value of power level is denoted by p_i^*, such that $p_i^* = \arg\max_{p_i} C_i(p_i)$. After calculation, the best possible improvement is broadcasted to notify its neighbors. Correspondingly, node i will make scholastic decisions of changing its current value. The probability of changing its current value can be calculated as:

$$\Pr(p_i \leftarrow p_i^*) = \begin{cases} \frac{C_p(p_i^*) - C_p(p_i)}{\sum_{j \in Adj(p_i) \cup i} C_p(p_j^*) - C_p(p_j)} & if C_p(p_i^*) \neq C_p(p_i) \\ 0 & if C_p(p_i^*) = C_p(p_i) \end{cases} \tag{4.15}$$

The above process of our proposed scheme is sketched in Algorithm 8.

Algorithm 8 DSBA

```
1: function DSBA
2:     Randomly choose a value
3:     while Not finished do
4:         Broadcast the new value
5:         Update the values of the neighbors
6:         Calculate the improvement C_p(p_j^*) − C_p(p_j)
7:         Broadcast the improvement
8:         Make the decision based on (4.15)
9:     end while
10: end function
```

4.1.6 System Analysis

QLRPCR also has similar data maintenance and updating mechanism compared with QLPCR. Besides the rate adaptation mechanism introduced, it has several distinct features as well.

- The first improvement is moving the interference price from power control

to routing. By doing so, the interference caused during routing process can also be taken into consideration. The example in chapter 1 can still be used to illustrate the significance.

Suppose we have the Q table of node A as:

Destination	Next hop	Q value
C	F	0.578
C	B	0.336
C	C	0.898
...		

As explained earlier, node A prefers node F as the next hop because of the higher Q value. However, extra traffic going through node F inevitably increases the congestion level. If there is a large amount of traffic passing through node F, a lot of flows are affected. On the other hand, although node B has a lower Q value, less flows may be affected. In this sense, node B may be a better choice.

- The second improvement is the multi-agent coordination system. As mentioned above, frequent change in topology may result in an unstable network environment. Therefore, we need a coordination mechanism to reduce the frequency of power level changes, meanwhile keep the benefit of power control scheme.

  Fig. 4.4 illustrate why a coordination system is important in MANETs. Suppose node A and B both notice node C is a good intermediate node for packet forwarding with a lower interference price. Without a coordination mechanism, both of them will increase their power levels and start to send packets to node C. As a result, the interference price of node C increases dramatically. Then, A and B may choose to decrease their power levels. As

a result, the network topology oscillates between these two states.

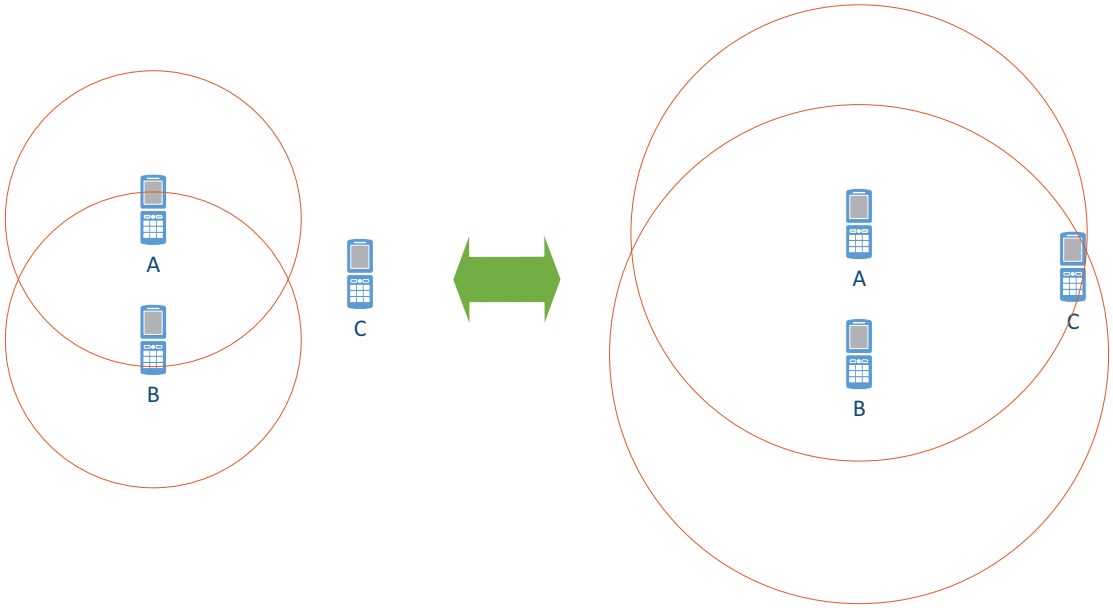


Fig. 4.4: Power control collision

The DSBA algorithm we proposed to address the problem is shown in Fig. 4.5. Before changing its power level, each node will broadcast the possible improvement of this change. Then each node will take the action stochastically so that a larger improvement can have a higher chance to change the power level.

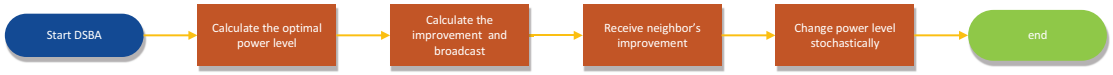


Fig. 4.5: Flowchart of DSBA

It can be seen that with the two new features, network efficiency and stability can improved. However, it inevitably introduces extra overhead for information sharing. Besides, the global-utility-based routing introduces a fairness problem: some nodes have to sacrifice its own performances to improve the overall network performances.

4.2 Simulation

Simulations are carried out over NS 3.12 using the built-in IEEE 802.11a MAC and Wi-Fi PHY models. The power consumption and modulation threshold setting follow the Cisco 802.11 a/b/g CardBus Wireless LAN card operating on 802.11a mode similar to [57]. The detailed parameters are listed below: We

Table 4.1: SINR Threshold for Different Data Rate

SINR threshold for 6 Mbps	6.02 dB
SINR threshold for 9 Mbps	7.78 dB
SINR threshold for 12 Mbps	9.03 dB
SINR threshold for 18 Mbps	10.79 dB
SINR threshold for 24 Mbps	17.04 dB
SINR threshold for 36 Mbps	18.80 dB
SINR threshold for 48 Mbps	24.05 dB
SINR threshold for 54 Mbps	24.56 dB

compare MRPCR against the original DSDV and TPCRA. DSDV uses a simple shortest path algorithm without a power control scheme. TPCRA constructs a topology by maximize the data rate for each link. Then a routing function considering data rate, path length and power level is used to choose the path. We evaluate our protocol with the original DSA model and also the simple extension from the chapter 3, QLRPCR. For each scenario, the delay, throughput and energy consumption are measured and analyzed. All values shown are averaged over 60 independent simulations and plotted with a 95% confidence interval. We evaluate four scenarios: the effect of varying node densities, mobility, flow densities and traffic loads. Note the scenarios in this chapter follows [57], therefore are quite different from previous chapters.

4.2.1 Node Density

In the first scenario, the number of nodes and flows are fixed with constant data rate. The simulation area is varied from 300m × 600m to 1km × 2km, in order to compare with [57]. Three sources and three destinations are placed at the leftmost and rightmost boundaries to elongate transmitting paths. Figure 4.6b shows that MRPCR has the shortest delay in most of the cases mainly due to the cross-layer scheme adopted in our system. DSDV preforms the worst because of its lack of power control and rate adaptation abilities. It can be seen that the delay in QLRPCR is quite small in a dense network due to its lower throughput: less packets to transmit cause less congestion. When the area gets larger, the power-controlled routing protocols begin to actively alter the transmitting power to construct better paths to destinations. That is also when MRPCR has significant advantages compared with other protocols. The delay performance of MRPCR remains the same in spite of whether the improved or the original DSA is adopted. The end-to-end delay is a measurement of accumulative latency of all links along a path. Therefore, the difference between the delay performances of these two stochastic processes are averaged out. TPCRA performs relatively better in a dense network while it suffers a significant performance downgrade when nodes getting further away. This is mainly due to its power control policy that requires constantly monitoring of all neighbors. If any of the neighbors cannot be reached with maximal transmitting rate, the node will choose its maximal transmitting power, in spite of whether these neighbors participate in packet forwarding. As a result, outliers heavily affect the power control decisions: those far-away nodes will force others to choose maximal power level although it may not be necessary.

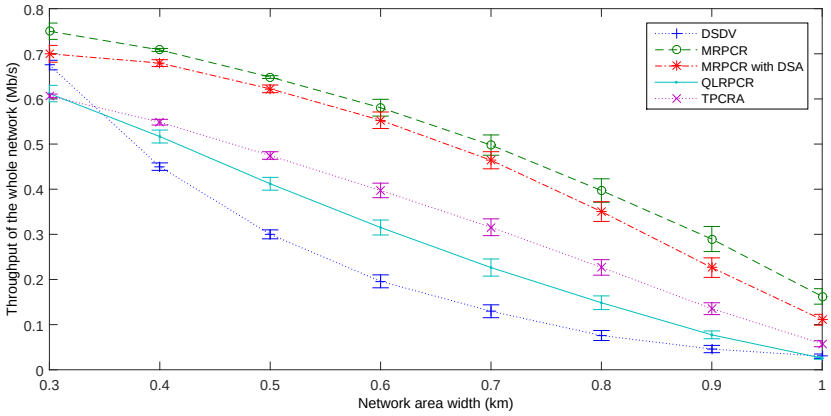
Figure 4.6a shows the end-to-end throughput performances. It can be seen

that MRPCR with the improved DSA has an observable improvement compared with the one with the original DSA. This is mainly due to the fact the original DSA requires a large number of iterations to reach the final output. Therefore, it introduces larger variation on the intermediate output. Such a variation is more significant in throughput because throughput is determined by the worst link along a path. DSDV can obtain good overall throughput in a dense network because of its smaller overhead. However, when the simulation area gets larger, this advantage is overturned by the need for an effective rate adaptation and transmitting power control scheme. Figure 4.6a also shows the trend that when the simulation area gets large enough all protocols will have the same performance because all nodes have to adopt maximal transmitting power and slowest modulation to maintain network connectivity.

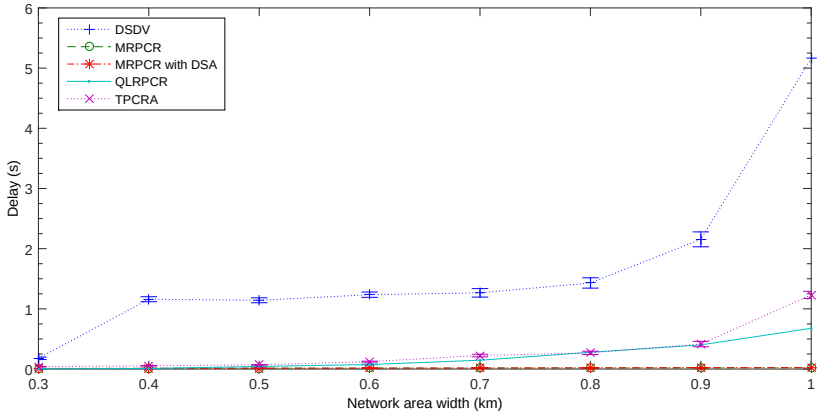
The average power consumption in Figure 4.6c is obtained by dividing the overall energy consumption of all nodes by the total simulation time and the number of nodes. Transmission power level and duration together determine the total power consumption. Generally speaking, higher throughput and slower data rate result in a longer transmission period. For example, DSDV has the largest energy consumption due to its high throughput when the simulation area is small. As the throughput decreases, the energy consumption decreases significantly because of the shorter transmitting duration. The energy consumptions of MRPCR and QLRPCR, on the other hand, have a peak value in the middle because the larger output power level overcomes the effect of the shorter transmitting duration.

4.2.2 Mobility

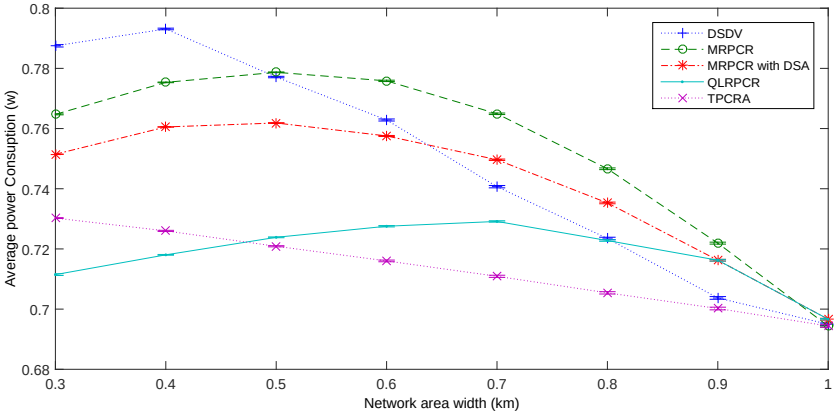
In this scenario, 50 nodes are randomly located at a 1200m $\times$ 600m area except for three fixed pairs of sources and destinations at the boundaries. The



(a) Throughput



(b) Delay



(c) Energy consumption

Fig. 4.6: QoS performance of power-controlled schemes under different node densities

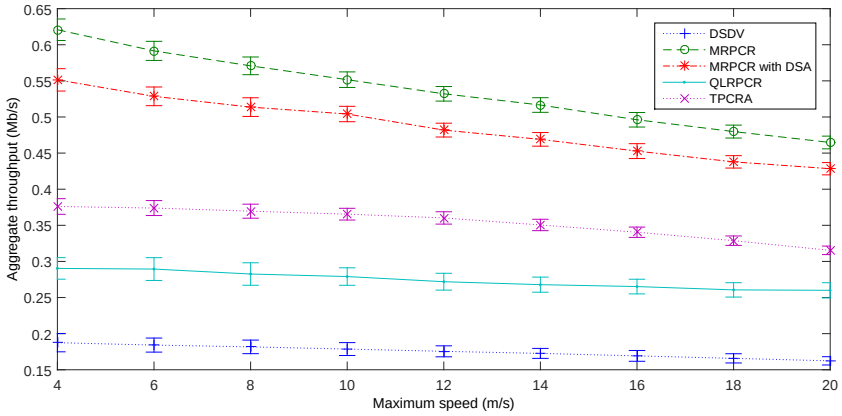
maximum speed of random waypoint is varied from 4m/s to 20m/s. It can be seen that the delay increases with mobility as shown in Figure 4.7b, due to the packet transmission failure caused by topology change. Such failure is handled by the retransmission mechanism in IEEE 802.11. If the retransmission attempt exceeds a threshold, the corresponding packet is dropped. Those dropped packets are not counted when measuring end-to-end delay performances. Therefore, latency degradation caused by mobility is not obvious. One exception is TPCRA, which almost triples its delay when the maximum speed increases from 4m/s to 20m/s. Higher power level in TPCRA can improve the throughput (Figure 4.7a) by increasing the transmission range. On other hand, it also reduces the chance of packet drop and consequently suffers more from retransmission mechanism.

MRPCR's throughput is mostly affected by mobility because each tends to use smaller power level to reduce interference level whenever possible. Therefore, as mobility gets higher, so does the packet drop rate. TPCRA, QLRPCR and DSDV have relatively stable performance because distance attenuation is less pronounced for them.

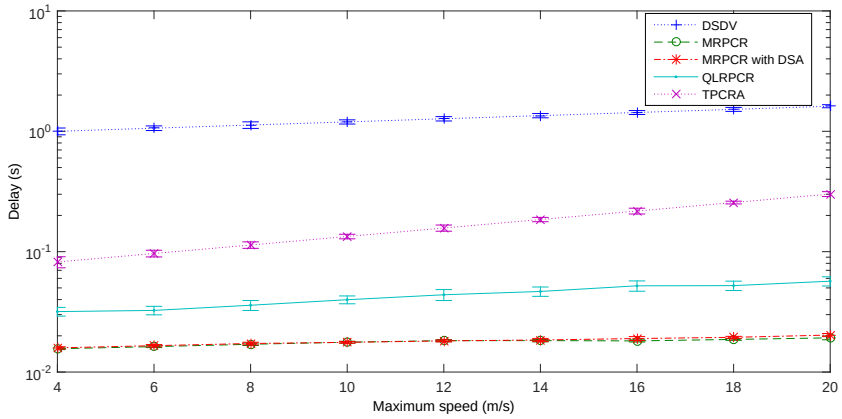
Fig. 4.7c shows the average power consumption is dominantly determined by the throughput except DSDV. Source nodes in DSDV have to keep sending routing request with maximal power level to establish new paths due to its extremely low throughput.

4.2.3 Traffic Load

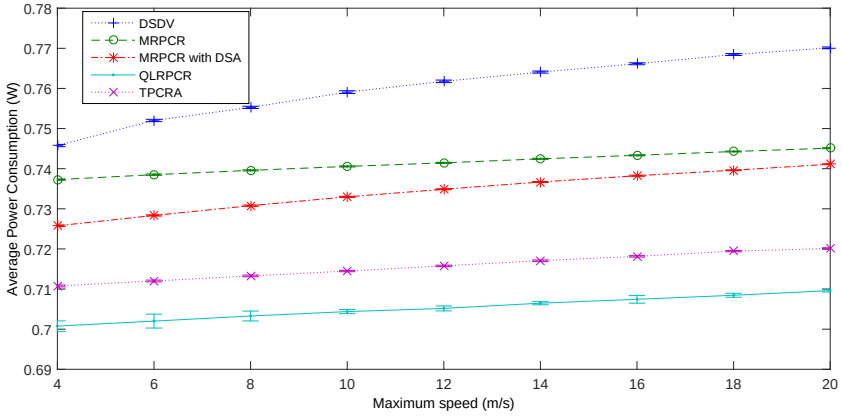
In this scenario, the traffic rate of each flow is varied from 128 kbps to 1280 kbps. MRPCR has a large jump in delay when the flow rate reaches 372 kbps, as shown in Figure 4.8b. When the throughput is approaching saturation point, the packet drop begins to increase, which results in lots of retransmission attempts.



(a) Throughput



(b) Delay



(c) Energy consumption

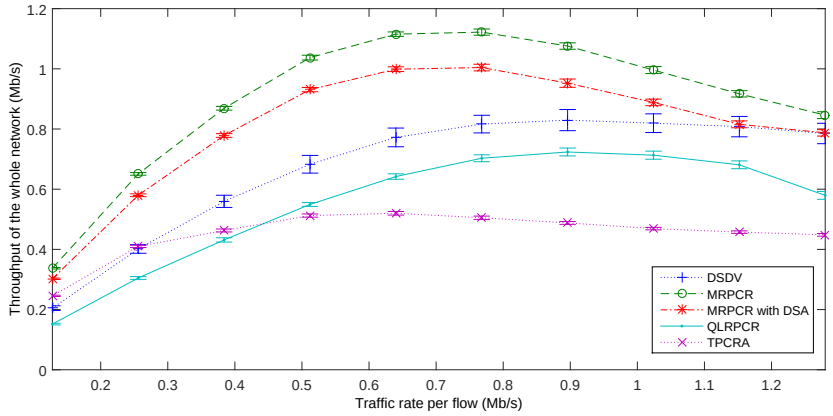
Fig. 4.7: QoS performance of power-controlled schemes under different mobilities

After the saturation point is reached, delay increment becomes relatively mild because of higher packet loss incurred. It is noticeable that although MRPCR uses delay as the only routing metric, in this scenario, its delay performance is worse than TPCRA and QLPCR when network becomes congested. The delay estimation model adopted in MRPCR is not an exact model. Nonetheless, it is still a good indicator of network conditions. Therefore, by using it, satisfactory delay and throughput performances can still be obtained (Fig. 4.8a).

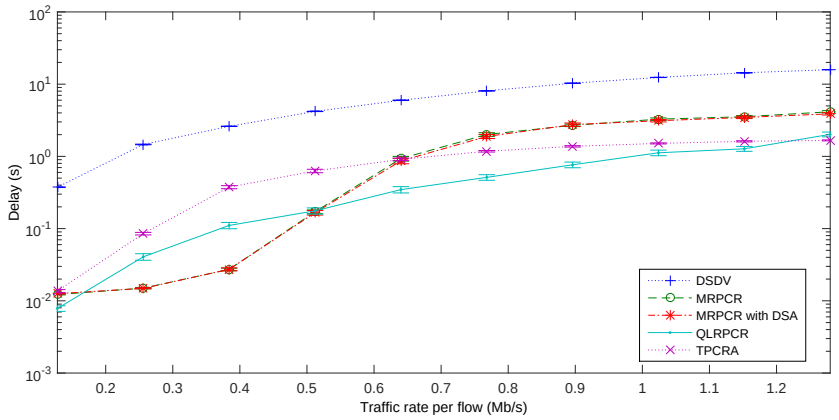
TPCRA performs well when the traffic load is low. However, its ignorance of interference results in significant throughput drop when the traffic load increases. It is also interesting to see that DSDV has better saturation throughput compared with TPCRA and QLPCR. It is because the shortest path algorithm may be efficient in a congested network. With smaller overhead and fewer intermediate notes involved, DSDV can mitigate the overall congestion level. On the other hand, the maximal transmission power introduces excessive interference for it as well. As a result, frequent retransmission attempts also elongate its average transmitting durations. Energy consumption follows the same pattern as in the last scenario showing throughput as a dominant factor (Fig. 4.8c).

4.2.4 Number of Flows

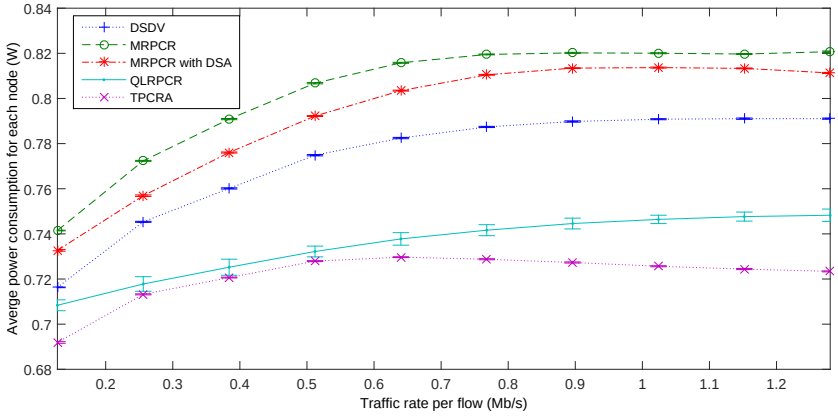
In this scenario, the number of flow is increased from 6 to 13, while the simulation area is fixed at 600 m $\times$ 800 m. Sources and destinations are randomly chosen from 50 nodes. The restriction of the 6 fixed nodes on the boundary is removed to ensure there are always 50 nodes freely moving in the simulation area. As a result, sources and destinations can be very close to each other at certain time intervals. It will inevitably produce a higher throughput and smaller delay compared with previous scenarios as shown in Fig. 4.9b and 4.9a. It can be seen



(a) Throughput



(b) Delay



(c) Energy consumption

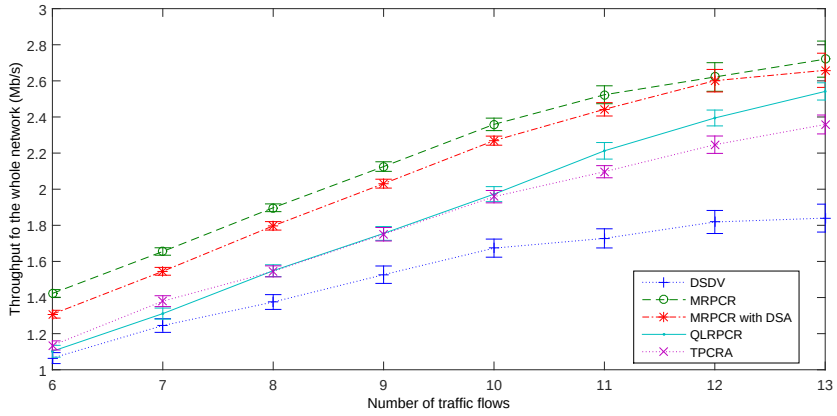
Fig. 4.8: QoS performance of power-controlled schemes under different traffic rates

that saturation is not reached with even 13 flows because of the high space reuse rate. MRPCR has the highest throughput due to its coordination mechanism. QLRPCR also explicitly considers interference in its metrics therefore achieves good throughput. TPCRA, on the other hand, lacks such a mechanism. As a result, it has better performances when the number of flows is small, but when the number of flows increases, the effect of excess interference becomes obvious which can also be observed in DSDV.

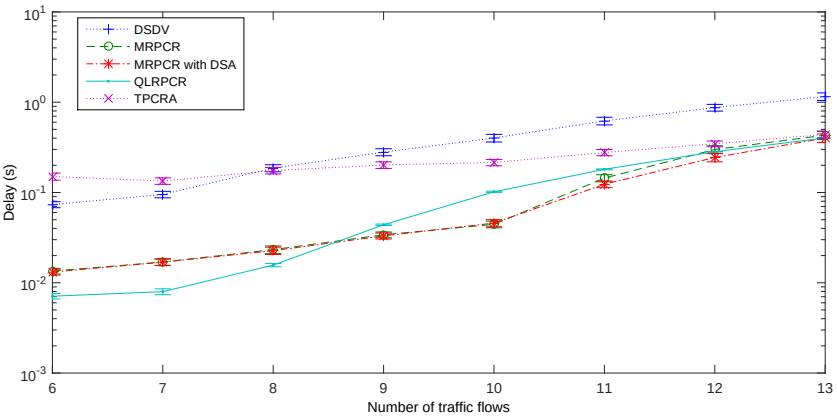
The end-to-end delay in QLRPCR is quite small when dealing with fewer flows (Fig. 4.9c). However, since it lacks of coordination mechanisms, when number of flows gets larger, congestion has a more severe impact on it. When the number of flows reaches 10, MRPCR with the original DSA has a smaller delay: If surrounded by multiple flows, decisions are more easily trapped in a local optimal point. The pure stochastic decision made by DSA will help the node jump out of local optimal point resulting in a better topology setting. TPCRA has relatively stable delay performances because of its aggressive power control policy.

4.3 Conclusion

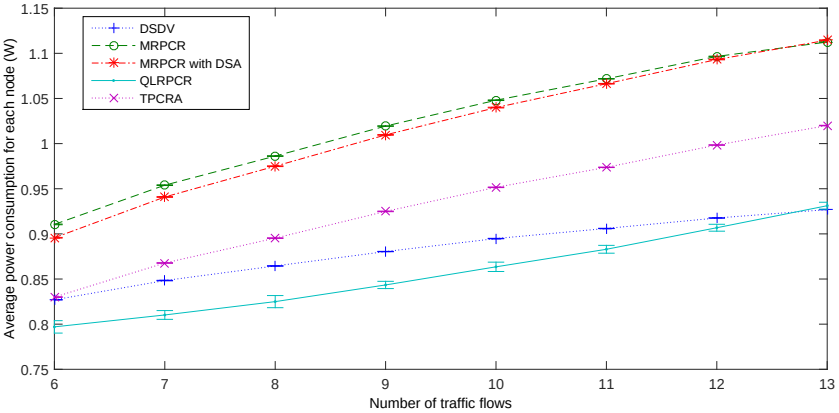
In this chapter, first we propose a simple extension of the chapter 3, with the rate-aware capability. However, in order to optimize the overall performance, we propose a power-controlled rate-aware routing protocol with multi-agent coordination enhancement. It adopts a customized Q-learning method that optimizes a global cost function based on a MAC-layer delay model. In addition, the multi-agent coordination mechanism stochastically prevents frequent changes of network to improve overall performances. Stimulation results show that our protocols can achieve a good delay and throughput performance with acceptable power con-



(a) Throughput



(b) Delay



(c) Energy consumption

Fig. 4.9: QoS performance of power-controlled schemes under different numbers of flows

sumption rate. Compared with the original DSA, the improved version almost always produces a better throughput. However, if surrounding environment is too complicated, the original DSA has a better chance to jump out of the local optimal point.

Chapter 5

Conclusion and Future Work

5.1 Summary

How to jointly optimize power, rate and routing in MANETs is the central problem in this thesis. To solve the complicated interactions between each other, we have provided a reinforcement-learning-based solution which dynamically learns and changes strategy based on the environment. We have also provided modifications to the traditional Q-learning method to overcome various problems in the context of MANETs. Step by step, we put routing, power control and rate adaptation into the framework of Q-learning.

The first part of our work is to apply the Q-learning method to the routing problem. We combine throughput, delay and SINR as the instant reward to guild the learning process. An automatic parameter tuning mechanism, which operates by observing the congestion level, is implemented to overcome the challenges introduced by multiple metrics. For different congestion level, different set of weights is adopted to bias the learning process. We also design a learning reset process to deal with the mobility. To overcome the routing loop problem, which is quite

common in the Q-learning-based routing schemes, we introduce a probing packets mechanism that periodically sends out probing packets that travel from source to destination while keep a record of all the nodes traversed. Once one node is visited more than once, the loop is detected. The probing packet then is forwarded again to notify all the node in the loop. To evaluate the proposed protocol, we compare the Q-learning-based routing protocol with the benchmarks by simulations. The simulation results show that our proposed protocol can perform well in different traffic loads, node densities, and link qualities.

The next step is to add power control into the Q-learning-based framework. The large action space formed by the power control and routing options prevents us from directly adding power level into the framework. To accelerate the learning process, a CSMA/CA model is introduced to simulate the reward so that all Q-entries can be updated at once. In the CSMA/CA model, we explicitly specify the rate of transmission attempts for each nodes to calculate the specific end-to-end-delay to each of the neighbors. The model also considers the hidden node problem by differentiating the neighbors of sender from the ones of receiver. A stochastic estimation method of mutual neighbors is design to reduce the communication overhead when determining the neighbors. The power control scheme involves multiple nodes in a network. Therefore, we measure the interference level by a weighted delay so that the trade-offs between the interference level and transmission range can be quantitatively calculated. Since only delay is used as the routing metrics, we also introduce a more efficient routing-loop prevention mechanism, which prevents self-addition into the routing entry. The simulation results show that our proposed protocol can perform well in a heavily loaded dense network with high mobility with comparable energy consumption.

Finally, we include the rate adaptation to form a complete solution. Instead of

extend from the previous power-controlled routing protocol, we choose to redesign the routing scheme to include the interference level to overcome the suboptimal solution illustrated in Braess's paradox. In order to properly measure the interference level, we make use of the diffusion model to obtain the derivate of the end-to-end delays. We also combine both DSA and DBA to provide a fast anytime solution for the multi-agent learning coordination problem. The simulation results show that our protocols can achieve a satisfactory performance in various scenarios. However, if surrounding environment is too complicated, the original DSA has a better chance to jump out of local optimal points.

In conclusion, by applying the Q-learning method to the cross-layer design, satisfactory QoS performances can be obtained. Since Q-learning is capable of learning from the environment, it is expected that learning nodes in MANETs can cope with various traffic conditions.

5.2 Future Work

The following possible improvements can be implemented in our protocol:

- Delay estimation is based on delay models. Therefore, a better delay model is also a possible way to improve our current work. Both the CSMA/CA and diffusion model make some unrealistic assumptions to simplify the process. Therefore, it is always possible to come out with a more precise model.

- Considering interference in routing introduces the fairness problem. A rotation-based scheduling can be added to ensure fairness.

- DCOP mechanism also has rooms for improvement. Although a complete algorithm is not applicable to MANETs, it is still possible to borrow some

ideas such as backtracking mechanism to come out with a hybrid algorithm that provide a locally complete algorithm.

There are also some potential directions we can continue our research:

- We use delay as the reward for path selection because it is additive. However, it is also possible to explore other solutions such as throughput. In graph theory, delay and throughput can be found by using the shortest path and maximal flow algorithms respectively. Therefore, in Q-learning-based routing schemes, it is also possible to obtain the maximal flow by sharing information among neighbors and modeling how each node affect the surrounding overall flow rate.

- Scheduling can be the next factor to be included in the Q-learning framework because it is also important to the QoS optimization in MANETs.

- Q-learning can be extended into higher level. Choosing delay as the metric of routing is predefined. However, it is possible to dynamically change the routing metrics based on higher level application requirement. For example, when user is watching streaming videos, the Q-learning process may choose throughput as the routing metrics whereas for instant messaging, delay is a better choice. In order to understand user's intention, we need more sophisticated AI knowledge representation such as ontology or natural language processing to make it feasible.

As for the QoS optimization problem, there are several possible solutions for MANETs as well:

- Caching is a possible way to solve the unreliable links in MANETs. When destination is not reachable, source may choose to send the packets to an

intermediate node which has higher possibility to reach the destination in the future. When the intermediate node finds a path to the destination, it will send the cached packets for the source.

- Network coding is also another solution to improve the network performance. Combining several messages together can effectively reduce the network traffic load.

Bibliography

[1] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Oar: an opportunistic auto-rate media access protocol for ad hoc networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 39 – 53, 2005.

[2] S. Biaz and S. Wu, "Rate adaptation algorithms for ieee 802.11 networks: A survey and comparison," in *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pp. 130–136, IEEE, 2008.

[3] R. Bellman, "On a routing problem," tech. rep., DTIC Document, 1956.

[4] L. Ford and D. R. Fulkerson, *Flows in networks*, vol. 1962. Princeton Princeton University Press, 1962.

[5] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968. a star search.

[7] H. Zimmermann, "Osi reference model–the iso model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.

[8] C. J. C. H. Watkins, *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[9] C. E. Perkins, *Ad hoc networking*. Addison-Wesley Professional, 2008.

[10] J. Zander, "Distributed cochannel interference control in cellular radio systems," *Vehicular Technology, IEEE Transactions on*, vol. 41, no. 3, pp. 305–311, 1992.

[11] G. J. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *Vehicular Technology, IEEE Transactions on*, vol. 42, no. 4, pp. 641–646, 1993.

[12] S. Ulukus and R. D. Yates, "Stochastic power control for cellular radio systems," *Communications, IEEE Transactions on*, vol. 46, no. 6, pp. 784–798, 1998.

[13] R. D. Yates, "A framework for uplink power control in cellular radio systems," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 7, pp. 1341–1347, 1995.

[14] L. Kleinrock and J. Silvester, "Optimum transmission radii for packet radio networks or why six is a magic number," in *Proceedings of the IEEE National Telecommunications Conference*, vol. 4, pp. 1–4, 1978.

[15] T.-C. Hou and V. O. Li, "Transmission range control in multihop packet radio networks," *Communications, IEEE Transactions on*, vol. 34, no. 1, pp. 38–44, 1986.

[16] T. ElBatt and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," *Wireless communications, IEEE Transactions on*, vol. 3, no. 1, pp. 74–85, 2004.

[17] C. Long, Q. Zhang, B. Li, H. Yang, and X. Guan, "Non-cooperative power control for wireless ad hoc networks with repeated games," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 6, pp. 1101–1112, 2007.

[18] X. Wu, X. Wang, and R. Liu, "Solving minimum power broadcast problem in wireless ad-hoc networks using genetic algorithm," in *Communication Networks and Services Research Conference, 2008. CNSR 2008. 6th Annual*, pp. 203–207, IEEE, 2008.

[19] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 404–413, IEEE, 2000.

[20] L. Correia, D. Macedo, A. dos Santos, A. Loureiro, and J. Nogueira, "Transmission power control techniques for wireless sensor networks," *Computer Networks*, vol. 51, no. 17, pp. 4765 – 79, 2007.

[21] E.-S. Jung and N. H. Vaidya, "A power control mac protocol for ad hoc networks," in *Proceedings of the 8th annual international conference on Mobile computing and networking*, pp. 36–47, ACM, 2002.

[22] J. Gomez, A. T. Campbell, M. Naghshineh, and C. Bisdikian, "Conserving transmission power in wireless ad hoc networks," in *Network Protocols, 2001. Ninth International Conference on*, pp. 24–34, IEEE, 2001.

[23] M. Kubisch, H. Karl, A. Wolisz, L. Zhong, and J. Rabaey, "Distributed algorithms for transmission power control in wireless sensor networks," in *IEEE Wireless Communications and Networking Conference Record*, pp. 558 – 63, 2003.

[24] S. Narayanaswamy, V. Kawadia, and R. S. Sreenivas, "Power control in adhoc networks theory architecture and implementation of the compow protocol," in *European Wireless Conference*, 2002.

[25] N. Pradhan and T. Saadawi, "Adaptive distributed power management algorithm for interference-aware topology control in mobile ad hoc networks," in *Proceedings 2010 IEEE Global Communications Conference (GLOBECOM 2010)*, pp. 6 pp. –, 2010.

[26] A. Kamerman and L. Monteban, "Wavelan-ii: A high-performance wireless lan for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118 – 133, 1997.

[27] M. Lacage, M. H. Manshaei, and T. Turletti, "Ieee 802.11 rate adaptation: A practical approach," in *ACM MSWiM 2004 - Proceedings of the Seventh ACM Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 126 – 134, 2004.

[28] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive mac protocol for multihop wireless networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, (Rome, Italy), pp. 236 – 250, 2001.

[29] Z. Li, A. Das, A. K. Gupta, and S. Nandi, "Full auto rate mac protocol for wireless ad hoc networks," *IEE proceedings-communications*, vol. 152, no. 3, pp. 311–319, 2005.

[30] J. Kim, S. Kim, S. Choi, and D. Qiao, "Cara: collision-aware rate adaptation for ieee 802.11 wlans," in *25th IEEE INFOCOM Conference*, pp. 2742 – 52, 2006.

[31] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, vol. 2006, pp. 146 – 157, 2006.

[32] R. Jantti and S.-L. Kim, "Joint data rate and power allocation for lifetime maximization in interference limited ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 5, pp. 1086 – 94, 2006.

[33] T.-S. Kim, H. Lim, and J. C. Hou, "Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, vol. 2006, pp. 366 – 377, 2006.

[34] A. Behzad and I. Rubin, "High transmission power increases the capacity of ad hoc wireless networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 1, pp. 156 – 165, 2006.

[35] C. R. Lin and J.-S. Liu, "Qos routing in ad hoc wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 8, pp. 1426–1438, 1999.

[36] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999.

[37] D. B. Johnson, D. A. Maltz, and J. Broch, *Ad hoc Networking*. Ed. Addison Wesley, 2001.

[38] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *Computer Communication Review*, vol. 24, pp. 234 – 44, 1994.

[39] C.-K. Liang and H.-S. Wang, "An ad hoc on-demand routing protocol with high packet delivery fraction," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004.

[40] M. Marina and S. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Ninth International Conference on Network Protocols*, 2001.

[41] M. Quwaider, J. Rao, and S. Biswas, "Neighborhood route diffusion for packet salvaging in networks with high mobility," in *IEEE International Performance, Computing and Communications Conference*, 2008.

[42] Y.-H. Chang, T. Ho, and L. Kaelbling, "Mobilized ad-hoc networks: a reinforcement learning approach," in *Proceedings of the First International Conference on Autonomic Computing*, (Los Alamitos, CA, USA), pp. 240 – 7, 2004.

[43] C. Wu, K. Kumekawa, and T. Kato, "A manet protocol considering link stability and bandwidth efficiency," in *International Conference on Ultra Modern Telecommunications Workshops (ICUMT)*, 2009.

[44] M. Soysal and E. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[45] D. Chetret, C.-K. Tham, and L. Wong, "Reinforcement learning and cmac-based adaptive routing for manets," in *12th IEEE International Conference on Networks (ICON)*, 2004.

[46] G. Santhi, A. Nachiappan, M. Ibrahime, R. Raghunadhane, and M. Favas, "Q-learning based adaptive qos routing protocol for manets," in *International Conference on Recent Trends in Information Technology (ICRTIT)*, 2011.

[47] N. M. Khan, Z. Khalid, and G. Ahmed, "Gradient cost establishment (grace) for an energy-aware routing in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, no. 1, pp. 1–15, 2009.

[48] C.-K. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 138 – 47, 2001.

[49] T. ElBatt, S. V. Krishnamurthy, D. Connors, S. Dao, *et al.*, "Power management for throughput enhancement in wireless ad-hoc networks," in *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, vol. 3, pp. 1506–1513, IEEE, 2000.

[50] N. Meghanathan, "On-demand maximum battery life routing with power sensitive power control in ad hoc networks," in *Proceedings of the International Conference on Networking, Systems, Communications and Learning Technologies*, 2006.

[51] F. Diamantopoulos and A. Economides, "Performance evaluation of power control routing for ad-hoc networks," in *12th European Wireless Conference 2006*, pp. 6 pp. –, 2006.

[52] X. Zhang, M. Liu, H. Gong, S. Lu, and J. Wu, "Pcar: A power controlled routing protocol for wireless ad hoc networks," in *IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2010.

[53] K. Tsudaka, M. Kawahara, A. Matsumoto, and H. Okada, "Power control routing for multi hop wireless ad-hoc network," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2001.

[54] H. Huang, G. Hu, and F. Yu, "A routing algorithm based on cross-layer power control in wireless ad hoc networks," in *5th International ICST Conference on Communications and Networking in China (CHINACOM)*, 2010.

[55] C.-F. Chou and H.-P. Suen, "Topology-control-based qos routing (tlqr) in wireless ad hoc networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, Institute of Electrical and Electronics Engineers Inc., 445 Hoes Lane / P.O. Box 1331, Piscataway, NJ 08855-1331, United States, 2006.

[56] S. Kwon and N. B. Shroff, "Energy-efficient interference-based routing for multi-hop wireless networks," in *Proceedings - IEEE INFOCOM*, 2006.

[57] D. Macedo, A. dos Santos, L. Correia, J. Nogueira, and G. Pujolle, "Transmission power and data rate aware routing on wireless networks," *Computer Networks*, vol. 54, no. 17, pp. 2979 – 90, 2010.

[58] C. J. Watkins, P. Dayan, R. S. Michalski, R. S. Michalski, G. Tecuci, J. G. Carbonell, and T. M. Mitchell, *Machine Learning*, vol. 8. Morgan Kaufmann, 1994.

[59] R. Bellman, "A markovian decision process," tech. rep., DTIC Document, 1957.

[60] R. Bellman, "Dynamic programming and lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

[61] A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[62] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1, pp. 560–564, IEEE, 1995.

[63] Nsnam, "ns-3." Available at `https://www.nsnam.org`, Accessed: 2015-06-30.

[64] Canonical, "Ubuntu." Available at `http://www.ubuntu.com`.

[65] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*, vol. 2. prentice hall PTR New Jersey, 1996.

[66] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New insights from a fixed-point analysis of single cell ieee 802.11 wlans," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 588 – 601, 2007/06/.

[67] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535 – 47, 2000/03/.

[68] E. Ziouva and T. Antonakopoulos, "Csma/ca performance under high traffic conditions: throughput and delay analysis," *Computer Communications*, vol. 25, no. 3, 2002.

[69] Q. Zhao, D. H. Tsang, and T. Sakurai, "A simple model for nonsaturated ieee 802.11 dcf networks," *IEEE Communications Letters*, vol. 12, no. 8, pp. 563 – 565, 2008.

[70] W. R. Gilks, *Markov chain monte carlo*. Wiley Online Library, 2005.

[71] L. J. Williams, "Technology advances from small unit operations situation awareness system development," *IEEE personal communications*, vol. 8, no. 1, pp. 30–33, 2001.

[72] K. Chadha, "The global positioning system: Challenges in bringing gps to mainstream consumers," in *Solid-State Circuits Conference, 1998. Digest of Technical Papers. 1998 IEEE International*, pp. 26–28, IEEE, 1998.

[73] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *Network, IEEE*, vol. 15, no. 6, pp. 30–39, 2001.

[74] G. Mao, B. D. Anderson, and B. Fidan, "Path loss exponent estimation for wireless sensor network localization," *Computer Networks*, vol. 51, no. 10, pp. 2467–2483, 2007.

[75] M. Haenggi and D. Puccinelli, "Routing in ad hoc networks: a case for long hops," *Communications Magazine, IEEE*, vol. 43, no. 10, pp. 93–101, 2005.

[76] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets*. Cambridge University Press, 2010.

[77] E. I. Pas and S. L. Principio, "Braess' paradox: Some new insights," *Transportation Research Part B: Methodological*, vol. 31 B, no. 3, pp. 265 – 276, 1997.

[78] H. Kobayashi, "Application of the diffusion approximation to queueing networks. i. equilibrium queue distributions," *Journal of the Association for Computing Machinery*, vol. 21, no. 2, pp. 316 – 28, 1974.

[79] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 2006.

[80] N. Bisnik and A. Abouzeid, "Queuing network models for delay analysis of multihop wireless ad hoc networks," *Ad Hoc Networks*, vol. 7, no. 1, pp. 79 – 97, 2009.

[81] A. Rogers, A. Farinelli, R. Stranders, and N. Jennings, "Bounded approximate decentralised coordination via the max-sum algorithm," *Artificial Intelligence*, vol. 175, no. 2, pp. 730 – 759, 2011.

[82] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "Adopt: Asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, vol. 161, no. 1-2, pp. 149 – 180, 2005.

[83] A. Chechetka and K. Sycara, "No-commitment branch and bound search for distributed constraint optimization," in *Proceedings of the International Conference on Autonomous Agents*, vol. 2006, pp. 1427 – 1429, 2006.

[84] A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization," in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 266 – 271, 2005.

[85] R. Mailler and V. Lesser, "Solving distributed constraint optimization problems using cooperative mediation," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004*, vol. 1, pp. 438 – 445, 2004.

[86] A. Gershman, R. Amnon, and M. Zivan, "Asynchronous forward bounding for distributed cops," *Journal of Artificial Intelligence Research*, vol. 34, pp. 61 – 88, 2009.

[87] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings, "Decentralised coordination of low-power embedded devices using the max-sum algorithm," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2, pp. 630 – 637, 2008.

[88] Y. Zhang and A. Mackworth, "Parallel and distributed algorithms for finite constraint satisfaction problems," in *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing (Cat. No.91TH0396-2)*, pp. 394 – 7, 1991.

[89] M. Yokoo and K. Hirayama, "Distributed breakout algorithm for solving distributed constraint satisfaction problems," in *ICMAS-96 Proceedings. Second International Conference on Multi-Agent Systems*, pp. 401 – 8, 1996.

List of Publications

1. K. Wang, W.-C. Wong, and T. Y. Chai, "A manet routing protocol using q-learning method integrated with bayesian network," in *IEEE International Conference on Communication Systems (ICCS)*, pp. 270-274, IEEE, 2012.

2. K.Wang, T. Y. Chai, and W.-C.Wong, "A q-learning-based power-controlled routing protocol in multihop wireless ad hoc network," in *International Conference on Networks (ICON)*, pp. 1-5, IEEE, 2013.

3. K. Wang, W.-C. Wong, and T. Y. Chai, "An adaptive delay-based power control and routing scheme," in *Signal Processing and Communication Systems (ICSPCS)*, 2013 7th International Conference on, pp. 1-7, IEEE, 2013.

4. K. Wang, T. Y. Chai, and W.-C. Wong, "An on-demand rate-aware joint power control and routing scheme," in *Wireless Communications and Networking Conference (WCNC)*, pp. 2126-2131, IEEE, 2014.

5. K. Wang, T. Y. Chai, and W.-C. Wong, "Routing, power control and rate adaptation: A q-learning-based cross-layer design," *Computer Networks, Elsevier* vol. 102, pp. 20 - 37, 2016.