MACHINE LEARNING BASED CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS

JEAN-YVES SAOSENG

NATIONAL UNIVERSITY OF SINGAPORE

2007

MACHINE LEARNING BASED CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS

JEAN-YVES SAOSENG

(*B. Eng. , Supelec, France*)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2007

Acknowledgments

I would like to express my gratitude to my supervisor, family members and friends who have helped me in one way or another during my study at NUS.

First, I would like to thank my supervisor, Dr Tham Chen Khong, for his direction and guidance given to me during the course of the project. He has been a source of motivation and encouragement in the course of this undertaking.

My warmest thanks to Computer Networks and Distributed Systems Laboratory members and its officer, Mr. Eric Poon, in making the laboratory such a nice place to work.

My study at National University of Singapore was made possible through graduate research scholarships. I am extremely thankful to NUS for the financial support.

April 11, 2007

Contents

Summary

The performance of wireless sensor networks strongly depends on the underlying transport protocol. The traffic characteristics in sensor networks are known to cause frequent congestion spots. In this thesis, novel adaptive methods in congestion control are explored.

In the first part of this thesis, a review of existing work in congestion control is given to highlight the congestion likelihood problem. Two means of congestion mitigation are employed depending on the sensing scenario. First, the regulation of channel contention is proposed for mitigation of transient congestion. Second, the packet generation rate is adjusted collaboratively to provide fairness and efficiency. Two artificial intelligence methods are investigated to solve these control problems. A first solution based on reinforcement learning is proposed to learn the policy which minimizes packet drop and unfairness. To this end, buffer overflows and greedy actions are punished with negative rewards. The SMART algorithm is then applied to maximize the long term average performance. The second solution is an inference technique called Min-Sum. The minimization of congestion is transformed into smaller coordination problems involving fewer variables. The interactions between sensors nodes are modelled in order to coordinate their control decisions.

The simulation results show that 15% improvement in energy efficiency is obtained over the recently proposed Fusion method. With a non-periodic workload, the proposed learning method provides privileged channel access to gateway nodes, making bandwidth available for higher aggregate throughput. With a periodic workload, the proposed

method still outperforms Min-Sum and Fusion in both fairness and efficiency. Although Min-Sum based methods allow accurate decision trade-offs, the message exchange is a limiting factor in the correctness of decisions.

This thesis shows that the congestion controller can learn the policy and hence does not require detection thresholds.

Publication

Saoseng J.-Y. and Tham Chen-Khong, "Coordinated Rate Control in Wireless Sensor Networks", accepted for 2006 IEEE International Conference on Communication Systems (ICCS), Singapore, 30 Oct - 1 Nov 2006.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents a brief overview of wireless sensor networks. The main objectives, contributions and the thesis structure are then summarized.

1.1 Background on wireless sensor networks

A Wireless Sensor Network (WSN) [1] consists of small micro-electronic devices with sensing, processing and communication capabilities. WSNs are intended to scale up to thousand of nodes and to cover large geographical areas. Sensors are scattered in the space of interest and then let to run with minimal human intervention. Wireless sensor networks promise a wide range of new applications such as habitat monitoring and target tracking.

Wireless communication suffers from radio interference, fading, high bit-error and collisions. Besides, wireless sensors communicate with little power and relatively simple protocols. Low energy consumption and reliable detection of the event are more significant attributes in WSNs. The packet is thus usually small and the transmission rates can be lower than one packet per second. Several types of workload are distinguished: event-driven, query-driven or periodic.

Applications running on wireless sensors have strict resource constraints. Sensor nodes are provided with little processing capabilities, small memory and low bandwidth. Moreover, they are often powered with batteries whose replacement is impractical. Since the radio component consumes energy the most [1], the communication protocols have to be energy efficient. For example, processing the data before transmitting can reduce communication. Implementing a sleep schedule is another way to increase the lifetime of the network.

The flows in WSNs are convergent and extend over multiple hops. One or several sinks are used for data collection. The collection usually has the pattern of a tree rooted at the sink. Intermediate nodes forward packets from the upstream branches towards the sink. In a WSN, congestion spots are frequently located near the sink and at the intersection of flows.

1.2 Motivation and objectives of the research

The channel is usually shared in WSNs and limited in capacity. Since the users of the network assign arbitrary sensing rates, the total load can exceed the capacity of the network. Therefore, congestion control is necessary to prevent packet drops. Wireless transmissions cost in energy and bandwidth. The dropping of a packet waste not only the last transmission but all the previous hops, hence diminishing the network energy efficiency. A congested network is not only inefficient but unfair: Congested paths are prone to packet loss. The network becomes unfair in presence of congestion because shorter paths have higher delivery ratio than longer paths.

Recent work [2, 3] suggests a three phases process: detection of congestion then its notification, followed by a rate adjustment. These methods entail two issues. First, the notification of congestion suffers from a random delay in contention based communications. Second, these schemes do not consider that different parts of network are more

likely congested than others.

The objectives of this thesis are multiple:

- To introduce adaptive methods for congestion control. The detection and rate adjustment parameters can be customized for a particular node to maximize the effectiveness of the control.
- To evaluate their efficiency and fairness for various traffic workloads.
- To establish the coordination between sensor nodes. Data collection is a collaborative task in WSNs. A sensor node may benefit when coordinating with its peers in order to mitigate congestion.

1.3 Main contributions

In this thesis, multi-agent technology is investigated to model the network of sensors, while reinforcement learning and inference solutions to the congestion control problem are explored. The agent controls either the *transmission* or the *generation* of data packets to reduce congestion and unfairness.

Two methods are studied to adapt the control decisions to the each sensor node's context:

- A model free method approximates the control policy by reinforcement learning. The control problem is represented as a Semi-Markov Decision Process (SMDP). Based on a feedback, the agent learns the action that maximizes an utility function.
- The inference method coordinates the agents' action to reach a global objective. The congestion control problem is formulated as a cost minimization. The costs are modelled with coordination functions. Messages are exchanged to determine the minimal cost action in a decentralized way.

The simulation results showed that the proposed methods can improve the energy efficiency up to 15% over the Fusion method[4] with periodic and non-periodic workload. They also provide fair collection from all sources.

1.4 Structure of the Thesis

The remainder of this thesis is organized as follows:

The next chapter surveys congestion control and avoidance schemes in WSNs. Design goals are also provided.

Chapter 3 presents the notification delay issue and its effect on congestion. The problem is tackled with an approach intending to reduce loss and also with another for fair allocation.

Chapter 4 gives introductory foundations to reinforcement learning. Then, the congestion control problem is translated as a learning problem where an agent is trying to interact optimally with its environment and other agents.

Chapter 5 presents the inference algorithm Min-Sum for coordinating agents. The theoretical background of the method is given, then interactions are modelled.

Chapter 6 presents the simulations and the results. Four proposed methods are compared to existing methods on a 19-nodes network.

Finally, Chapter 7 concludes the thesis, highlighting the major contributions.

Chapter 2

Literature review

This chapter presents the state of the art in congestion control for wireless sensor networks. The main design points of a congestion control scheme are laid out. The existing methods can be classified in either congestion avoidance or congestion control. Most of existing work combines several methods to increase both efficiency and fairness.

2.1 Introduction

2.1.1 Congestion in sensor networks

In wireless sensor scenarios, the sink is not always within direct transmission range. Intermediates nodes forward packets and also generate packets. The bandwidth allocation is especially complex in a multi-hop ad-hoc network. Given that the channel capacity is limited in capacity, sensor nodes must adapt their transmission and their packet generation rate. If the source nodes near the sink originate too much bandwidth, little is left for distant nodes. Conversely, if the distant nodes originate important traffic, the nodes downstream will drop packets by congestion, wasting the previous effort to forward them. Moreover, transient congestions are frequent in wireless networks, because

the channel condition varies with time.

Congestion also occurs when events are reported collectively. For example, sensors tracking the same event will simultaneously create packets. The channel can become congested since several nodes transmit within a very short period. The buffers eventually overflow when the packets propagate in burst towards the sink.

Nodes near the sink send a lot of packets because they route the traffic from the entire upstream spanning tree. Since each hop has a congestion probability, a path with more hops suffers from more loss. Therefore congestion leads to unfairness. Distant sources suffer from greater loss than closer sources whose packets are collected in larger proportion.

Congestion control intends to limit the effects of congestion such as packet drops and delay. A fair allocation of bandwidth is desirable as it ensures an uniform sensing coverage. In WSNs, energy efficiency is important and few seconds of delays are tolerated in most applications.

Physical errors can be the first cause of congestion [4]. Radio transmissions have effects beyond the reception range. Two wireless sensors may not able to communicate directly, but can mutually interfere each other as shown in Figure 2.1. Interferences lead to high error and collision probability. Congestion collapse can occurs even with large buffers since congestion may cause corrupted transmissions due to interferences.

2.1.2 Design criteria in congestion control

In WSNs, several criteria should be taken into consideration in the design of a congestion control scheme:

1. Burst management: congestion occurs not only with a periodic workload, but also with an event-based workload.

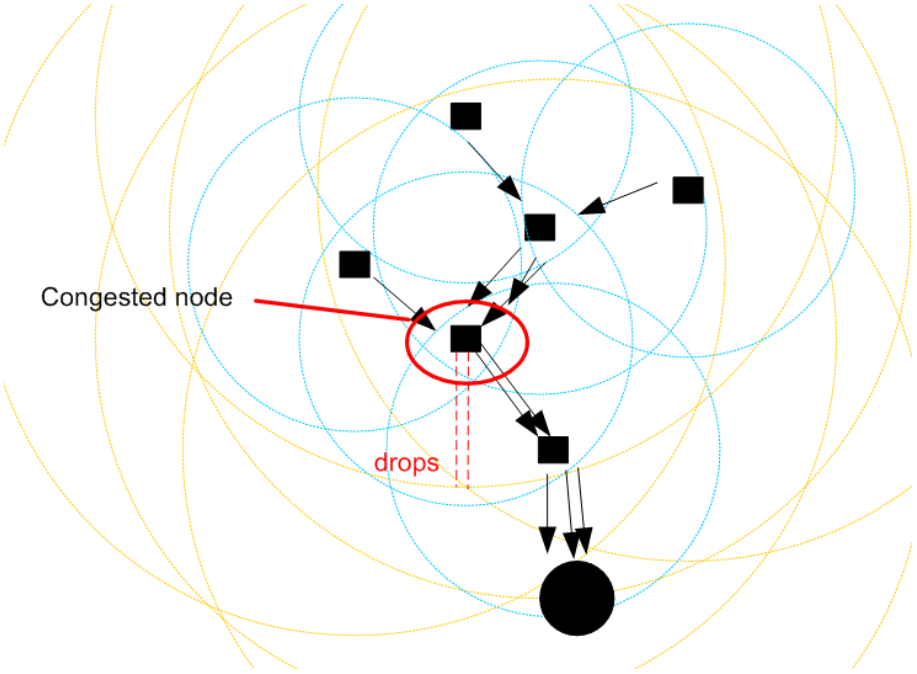

Figure 2.1: Wireless sensor network with a congested node. The transmission and interference range are shown with circles.

2. Stable response: A fast detection and response to congestion permits more energy savings. However, with a random channel access, a delay is expected in the propagation of control messages. The control system should avoid response to temporary perturbation that could cause oscillation.

3. Load diversity: Since the report rate is specific to the application, the congestion control scheme should give consistent performance in light and heavy load. If the offered load exceeds the network capacity, the scheme must be able to minimize congestion. In addition, some sources may be more important than others, thus the scheme should distinguish the weight of packets.

4. Robust in imperfect wireless environment: Wireless links can lose up to 20% of packets due to errors. Besides, topology changes can occur in real deployment.

5. Adaptability: Self configuration of the algorithm is an useful feature since sensors

are deployed in large numbers. The congestion control scheme may adapt to the particular environment of a node.

6. Simple: The overhead induced by control signals or by processing is kept low in order to not hinder the main application.

Finally, the scalability of the scheme must be indisputable.

Beside conventional quality of service metrics like drop rate, throughput and delay, other design criteria are relevant to wireless sensor networks:

- *Energy efficiency* is essential since congestion affects the lifetime of the network. Packet drops waste the energy spent in the previous hops. Collisions and channel errors are additional energy waste because of retransmissions. The efficiency expressed in J/packet is proportional to the number of packets collected at the sink.

- *Fairness*: The utility of the information collected is optimal when the delivery is perfectly balanced. In a congested network, far sensor nodes deliver packets with difficulty. As a result, fairness diminishes because the collected information mainly comes from nodes next to the sink.

- *Reliability*: The application may define a minimum reliability which requires a certain delivery ratio. This metric is also called the *perceived application fidelity*. In general, sensing applications do not require end-to-end packet reliability since the information is redundant.

In the sensor network literature, congestion control schemes differ in the way of detecting a congestion, of signaling it and in the way of adjusting the rates. Two congestion

control strategies exist: congestion control and congestion avoidance. A congestion control mechanism prevents abrupt congestion collapse by ensuring throughput and fairness deteriorate gracefully with the load. Congestion avoidance techniques adjust the rates after congestion is detected.

2.2 Congestion Avoidance

Congestion avoidance mechanisms have been reported since the development of MAC protocols in sensor networks [2]. They were initially designed to improve the network reliability, by preventing packet drops. New issues related to congestion were raised and several techniques were proposed such as: buffer based detection, fair rate allocation and prioritized channel access. Various types of rate adjustment are reported: heuristic, exact, or based on the current congestion level. This section presents the several congestion indicators and the rate adjustment techniques frequently met in the WSN literature.

2.2.1 Congestion Detection

Congestion detection is estimating the probability of packet loss or delay. In conjunction with mitigation techniques, an accurate detection can anticipate a growing congestion and prevent the overflows of buffers. The congestion indicators have an essential role to avoid congestion. In addition, the rate adjustment can be dynamically adjusted according to the congestion levels to enhance response and stability.

Buffer occupancy is the prevalent congestion indicator [3, 4, 5, 6, 7, 8, 9, 10] since measuring the queue length is fairly simple. Packets can come from upstream nodes faster than they leave to the downstream node especially with contention based com-

munications. These packets in excess are stored in the buffer to improve reliability. Nevertheless the reliability cannot be perfect since when the buffer is full, arriving packets are dropped and definitely lost. An empty queue suggests a low contention level. A queue nearly full indicates a very likely congestion. Nevertheless, the buffer size affects the pertinence of buffer based detection. A large buffer takes time to overflow and thus a transmission imbalance could go undetected. Conversely, high occupancy does not imply channel congestion if the queue is decreasing.

A threshold is a straightforward mean to detect buffer congestion [3, 4, 10]. Congestion is implied if the buffer occupancy is above the threshold,. Chen et al. proposed a buffer management scheme [9] and studied the effect of hidden terminals on the congestion detection. The threshold is decreased with the number of child nodes. If k child nodes are contending for the same buffer space, k times less residual buffer is advertised. IFRC [7] uses multiple thresholds to assess congestion. The first threshold detects congestion for an increasing queue, the second for a decreasing queue. Large buffers equip sensors in IFRC [7] and can make congestion detection difficult. By no surprise, IFRC decreases dynamically the threshold until the buffer length stabilizes. In ESRT [5] and PCCMAC [10], the future buffer occupancy is estimated and overflows can be predicted. Prediction methods have the advantage of not using a threshold, avoiding the inefficiency caused by its estimation. Nonetheless, they assume that the arrival and service rates are constant in a close future, which is not always verified on a wireless medium.

Buffer congestion often has its root in a congestion of the channel. When the channel is at its maximum capacity, new packets are accumulated in the buffer until transmission is possible. To obtain the channel load, the sensor 's radio must be on and receiving. To preserve energy, Coda method [3] performs channel sampling only when the buffer contains packets.

Congestion is implied from packet loss in ARC [2]. Coda [3] additionally infers congestion from loss of ACK on the path between source and sink. Such detection method can mistake channel error with buffer congestion, and reduce the throughput unnecessarily. The arrival and service rate are used as congestion indicator in [8, 10]. This method is conservative as a node may lack opportunity to transmit not because congestion but of random accesses. Like the channel loading, these indicators are calculated with an average (EWMA). The determination of the time window in the average possibly affects the responsiveness of the entire scheme.

Recent studies [4, 3] showed that buffer occupancy is sufficient to detect congestion and is more accurate than channel loading. Proactive method like buffer occupancy can reduce the size of hot spots with a hop-by-hop flow control [4].

Small buffers are advised in sensor networks because the occupancy level indicates only the recent congestion and not the cumulated congestion. Large buffers produce longer delays that are penalizing for real time application. In most works using a threshold [4, 3, 6], the threshold value is determined *from intuition* and approximately set to 0.75. A common single threshold leads to inefficiencies because each of sensor node has a different congestion inclination. In most works, the threshold value is based on heuristics rather than rigorous analysis.

2.2.2 Congestion Notification

Upon congestion detection, the relevant nodes are notified in order to reduce the traffic going through congested nodes. Closed loop control is an end-to-end flow control between a source and the sink. Open loop control is a hop-by-hop flow control performed between adjacent sensors nodes.

Hop-by-hop

Hop-by-hop flow control has the advantage of scaling well with the network size since communications are only between adjacent nodes. Therefore a processing required for congestion control is moved from the base station to local sensors. The congestion notification propagates in the direction opposite to the data traffic and is also called backpressure. As the source nodes causing congestion can be located far from where congestion occurs, the backpressure messages propagate over several hops. Two situations are present: backpressure travels over non-congested sensor, thus the response to congestion is short. In the other case, congestion grows neighbor to neighbor until the target sensor is regulated. The reactivity of backpressure is very slow in this former case. It employs buffering to mitigate transient congestion.

Congestion is notified with the help of a congestion bit in Coda and Fusion [3, 4]. Real valued information will allows more accurate adjustment [7, 10]. In the explicit congestion notification (ECN), a control message is sent. The communication overhead of ECN motivates implicit congestion notification (ICN) which uses normal data packets to carry congestion information. By taking advantage of the broadcast nature of transmissions, the target sensors receive the control information by overhearing it. The assumption of overhearing unaddressed packets is common in wireless sensor protocols.

End-to-end

End-to-end congestion control happens between the source and the destination of a flow. The destination is responsible for loss detection and loss recovery. The sink feedbacks the sources after a congestion is inferred upstream. The congestion signals first need to propagate towards the sink. Then, rate control messages return to the sources causing congestion. In WSNs, end-to-end control will have a slow response to congestion be-

cause signaling is over several hops. Even though the sink is less resource limited, the acknowledgment messages take bandwidth in the entire network and has no guaranty of reaching the targets.

Event-to-Sink Reliable Transport (ESRT) [5] tags the packet with a congestion notification bit (CN) when the buffer is nearly full. In response to CN bits, the sink reduces globally the reporting rate according to an empirical control law. It assumes that the sink can broadcast the control message directly to all the sensor nodes. ESRT reduces the global rate until all congestion spots are cleared. The most congested source is restricting the whole network. Since congestion hot-spots can be transient and localized, a number of sources stay at a conservative rate.

CODA [3] applies both end-to-end and hop-by-hop flow control. With a closed loop multi-source regulation, the sink asserts congestion over several sources. When the rate is above a threshold, the sensor keeps its current rate only if it receives ACK messages periodically. The sink computes the reporting rate and limits the ACK messages sent to sources. Coda does not consider fairness in these regulations.

Hop-by-hop congestion control is attractive because it is effective and scalable. In terms of communication overhead, the end-to-end notification is costly without broadcasting or multi-casting. In terms of responsiveness, hop-by-hop flow control handles transient congestions the fastest. Moreover, some rate control strategies can provide efficient control of persistent congestion.

2.2.3 Rate Control

The mitigation of congestion is achieved by reducing the rate at which packets are generated. A rate control can be applied at different communication layers. However, a simple extension of the efforts in traditional network to WSNs meets several difficulties.

First, the channel quality and bandwidth are time-variant. Second, the nodes causing congestion are not obvious to find in the spanning tree.

AIMD: Additive Increase Multiplicative Decrease

The AIMD control augments the rate with a constant α until the signs of congestion appear; the rate is then cut by multiplying with $\beta < 1$. A larger α tends to a more aggressive channel contention. The choice of β determines the penalty at transmission failure.

CODA [3] decreases the sensing rate when the parent node is congested. The Fusion scheme [4] applies a stop-and-go rate control. The outgoing transmissions are completely stopped until the parent node clears its congestion. Such complete stop prevents buffer overflows more effectively than a simple decrement. IFRC [7] is a scheme that uses AIMD in juxtaposition with congestion sharing to allocate rates fairly. The stability of the scheme strongly depends on a heuristic which determines the AIMD parameters. ARC [2] maintains two independent sets of α and β to guarantee the fairness of the originating traffic with the route-thru traffic.

AIMD is actually a heuristic method. The available bandwidth at a sensor node continually changes in reason of interferences, multi-path fading and burst packets. AIMD based methods performs continually rate adjustment to adapt to a dynamic environment. The oscillation of the rates is inherent to AIMD as it cannot reach an equilibrium. Examples of oscillation are found in [6, 9, 7].

One difficulty of AIMD rate control is the determination of the exact rate reduction in response to congestion. An abrupt decrease of the transmission rate causes the backpressure to propagate over several hops, intensifying the oscillation. The additive increase can lead to inefficient utilization because resources are not fully used when avail-

able. IFRC for instance achieved only the 60% of maximum throughput. Even though some heuristics can improve convergence by tuning the AIMD parameters, adapting directly to the optimal rates would be more energy efficient than a slow converging rate control.

Exact rate adjustment

The general concept is to calculate the upper bound of the rate according to criteria like fairness. A fair share rate for each flow is $1/N$ where N is the number of source nodes in the upstream subtree. CCF [6] limits the packet generation rate r by C/N where C is the available capacity at a sensor node. Similar constraints are taken in PCCMAC [10].

Within a subtree, the generation rate cannot exceed the share just calculated. Since any subtree is at least included by the entire spanning tree, the rate of a source is determined with the smallest rate on the path from the source to the sink. To achieve a fair bandwidth allocation, the most loaded intermediate node imposes the maximum fair share of the bandwidth per source. The optimal fair rate of the network is equal to the minimum of the calculated fair share. This rate is propagated in the network by using a second constraint [6, 7]: the local generation rate r_i has to be inferior to parent's rate i.e. $r_i < r_{parent(i)}$. A node thus gains authority to control the rates of the all descendent nodes. The control of the upstream nodes is done recursively, simplifying the flow control. A minor inconvenient is that the sink has to transmit dummy packets at the maximum allowable rate as it is a parent node as well [7].

IFRC [7] considers in addition congestion sharing to achieve fair rate allocation. Nodes interfering with a congested node are throttled down although they are not direct neighbor in the tree. Potential interferers are the nodes whose flows share the channel with the flows of a congested sensor. The rate is decreased for nodes with a congested descendant and also for all potential interferers.

Channel contention

With contention based protocols, rate control can be unreliable because the maximum rate or available bandwidth is time-variant. A node is not able to transmit at a wanted rate r_i especially when the channel utilization is near saturation. The transmission rate can be changed dynamically through the back-off intervals in CSMA based protocol.

The rationale in controlling the contention is to give higher channel priority to gateway nodes which forward a lot of packets. To reach a given transmitting rate, the contention window must be correctly adjusted in function of neighbors' contention level.

PCCMAC [10] calculates virtual rates and the back-off window which allows the wanted rate. It proposes a virtual rate estimator since global information is required in the calculation. Congested sensor nodes in Fusion [4] uses shorter back-off to clear their buffer quickly. A prioritized access also accelerates the propagation of the backpressure. Woo et al. [2] point out that a random delay introduces a phase shift which can reduce channel congestion.

2.3 Congestion Control

The previous section has presented some solutions when congestion has been declared. This section surveys preemptive approaches to prevent congestion in WSNs. The delivery of packet is fair if the sink can receive roughly same number of packets from each sensor. If each node is to minimize loss without a reference, uneven magnitude of control leads to an unfair collection of packets.

2.3.1 Traffic shaping

Traffic shaping provides mechanism to control the traffic being sent in the network. Leaky bucket and token bucket are simple traffic shaping techniques to maintain a rate uniform.

The rate limitation suggested in Fusion [4] uses a token bucket to meter the traffic introduced in the network. It limits the source to generate at the same rate as all of its descendent. A source i generates packet at rate equal to $1/N(parent(i))th$ of the outgoing bandwidth of i's parent node. In terms of packets, for N packets send by the parent node, one packet is generated locally. Because burst of packets are common in WSNs, a token bucket is used. One token is generated for every N packets routed through the parent and another is spent to generate a packet. It entails promiscuous hearing.

Rate limiting removes greedy sources, and improves simultaneously fairness and efficiency. Nevertheless, traffic shaping does not accommodate well traffic from correlated events. Imbalance between incoming and outgoing traffic are not solved because the token bucket allows burst transmission. Also, since the N packets counted are not necessarily from unique sources, rate limiting is not perfectly fair when there are retransmissions. Lastly, the sink needs to send packets additionally so that its child nodes can shape their traffic.

2.3.2 Queue Management

With a first-in first-out queue discipline, the sources with a large traffic are advantaged because the buffer does not differentiate the incoming packets. Since it is beneficial for a source to send as many packets as possible, intermediate nodes become congested when the same strategy is taken by all nodes.

Packets can be differentiated according to their origin or their weight. PCCMAC [10] uses a Start time Fair Queue (SFQ) for fairness. Managing distinct queues can improve

fairness. One queue per child node is implemented in CCF [6]. The probability of servicing (PS) a queue is proportional to tree size N of the node linked to this queue. The PS mechanism allows packets to have equal probability of reaching the sink. Nevertheless PS requires work conserving queues which is hardly verified in practice.

The method was improved with an Epoch-based Proportional Selection (EPS). The node serves N packets in each epoch. Specifically, each queue serves a number of packets equal to the subtree size. Since EPS differentiate packets, it is robust with an imperfect channel. Moreover, congestion in any branches of the network will cause a decrease throughout all others parts of the network.

2.3.3 Adaptive Routing

Most of the transport protocols in WSNs consider single path routing. Load balancing can be achieved with multiple paths routing. Since a node can have several parent nodes, backpressure and the rate adjustment become problematic. It has to determine the rate that verifies simultaneously the constraints of several forwarding nodes, and in the same time has to maintain fair sharing of bandwidth. Multiple paths routing introduces a trade-off between delay and energy. Alternative routes can be less congested but need more hops. Furthermore route update messages may increase energy consumption.

Cross-layer optimization in wireless sensor network is still an open problem. Redirecting the packets to less congested nodes is the approach taken by Siphon [11]. Siphon employs virtual sinks to redirect traffic and mitigate congestion in the area where the traffic converges. But Siphon's virtual sinks are powerful nodes which communicate to the sink directly with a second radio.

2.4 Conclusion

The unique aspects of wireless sensor networks highlight the need for a fair and efficient congestion control scheme. The ideal transport scheme would remain energy efficient, fair and with good perceived application fidelity under heavy load conditions. A summary of the existing works is given in Table 2.1.

Buffer occupancy detects congestion sufficiently well to use as a base to improve. The use of a detection threshold is not convincing, and thus other methods are explored in this thesis. Adaption is one characteristic that this thesis explores.

Following most of the existing works, the proposed method in this thesis detects congestion from the buffer occupancy and mitigates with rate and contention control. Traffic shaping is a simple and conclusive technique to provide fair bandwidth allocation. Since rates are not exact in wireless environment, a control at the packet level answers better to the need of fast response and efficiency. In the following Chapters, learning and inference techniques are explored to find control strategies.

Table 2.1: Summary of existing congestion control schemes in WSNs

Scheme	Detection Method	Efficiency	Fair	Response
ARC [2]	Loss	Drop	Yes	Slow
CODA [3]	Static Buffer+ Channel	Drop	No	average
Fusion [4]	Static Buffer	Drop	Simple fairness	Fast
ESRT [5]	Predictive Buffer	No drop	Yes	Very slow
IFRC [7]	Dynamic Buffer	No drop	Max Min fair	Very slow
LWBM [9]	Dynamic Buffer	No drop		Very fast
CCF [10]	Predictive Buffer	No drop	Simple fairness	Fast

Chapter 3

Link Flow Control Problem

In this chapter, the challenges of congestion control in WSNs are presented. Sensor networks exhibit non-uniform congestion likelihood and coordinated behavior. Moreover, a shared wireless channel entails contention delay. An agent model is provided with detection and control features. The control of either the transmission rate or the packet generation rate is proposed to mitigate congestion.

3.1 Problem Statement

The previous chapter raises the difficulty in effectively detecting and mitigating congestion. The congestion properties are actually specific to a sensor node. The effectiveness of a detection method varies among nodes. A scheme can thus adapt its parameters to minimize congestion detection errors. Congestion has three characteristics which motivate the adaptation of a congestion control scheme:

First, congestion is unpredictable in location and size. The wireless environment is a factor of congestion. Multi-path interference, for instance, depends on the geographical location and limits the effective throughput. The maximum capacity of the network

is also determined by its topology. Furthermore, sensors nodes are randomly scattered making the traffic pattern unpredictable.

The size of congestion spots varies not only with the bandwidth but also with the number of child nodes. The congestion likelihood increases with the degree of the node: if the channel access probability is equal for all nodes, one node receives as many packets as child nodes between two transmissions.

Even the location in the network influences the congestion. Congestion causes little degradation when taking place deep in the network. On the contrary, a congestion near the sink causes serious drops and unfairness.

Second, in CSMA based communications, congestion notification is affected by the *contention delay* which increases the latency of rates updates. Hop-by-hop flow control is prevalent in WSNs. Messages are propagated to inform relevant sensors on the occurrence of congestion and the necessary rate adjustment. A sensor node can notify a congestion in average after half of its neighbors have transmitted if the channel capacity is almost reached.

While a congested parent node contends for the channel, a child node can transmit a packet that causes its parent node's buffer to overflow. Therefore, a slow response to congestion is detrimental to the network as more packets are dropped. Figure 3.1 illustrates the delay of notification.

To reduce the effects of contention delay, different strategies were suggested in the literature. A prioritized channel access is given to congested nodes in Fusion [4]. Large buffers [7] prevented the overflow resulting from uncontrolled queue rises. A buffer reservation scheme was suggested in [9]. The buffer occupancy methods [3] notify upstream nodes before the buffer is completely full. However the outcome of actions has uncertainties related to randomness of channel access. None of the existing works has considered the congestion control as a quantifiable and collaborative task in a stochastic

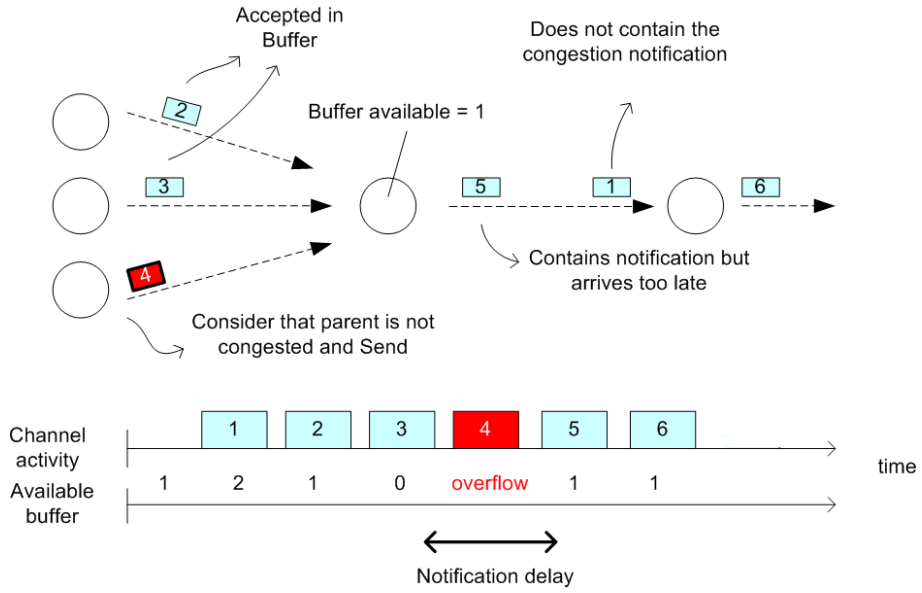

Figure 3.1: Delay in congestion notification: an intermediate node forwards packets for three child nodes. The packets are numbered according to the order of transmission. Child nodes stop transmitting when they hear the parent notifying congestion. However, random contention prevents immediate notification. During this delay lapse, the child nodes keep transmitting at the high rate, causing buffer overflow.

environment.

Lastly, congestion is the result of joint actions in addition to individual actions. The congestion control is a problem where sensors have to coordinate between themselves. Without communicating their knowledge, the transmission strategies may be in conflict and lead to worse congestion.

For a given sign of congestion, the congestion detector determines the corresponding congestion likelihood and the corresponding control action. Agent technology can overcome these problems. The autonomous nature of agent would be able to represents these sensor nodes making rational decisions to solve the congestion.

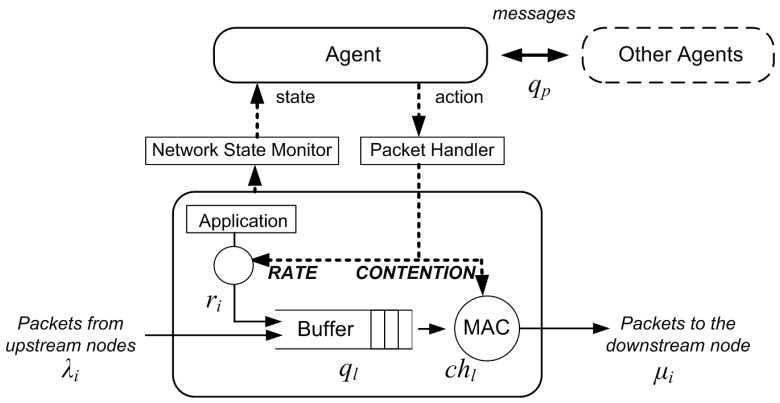

Figure 3.2: Link Flow Control model

3.2 Agent Model of the sensor node

Modeling a WSN as a multi-agent system [12] is well-founded because a wireless sensor has computation and communication capabilities. A global control is not robust since a failure of the deciding agent would be detrimental to the global performance. On the contrary, the use of multiple autonomous agents provides flexibility and adaptability. In the context of WSNs, agents are software entities with the same communication protocol. Reusability of the code is important when hundreds of sensors are deployed.

An agent aims for a goal by performing three tasks iteratively: observation of the conditions of the environment, reasoning to interpret the observations, and then action. The locally observed environment can be affected by other agents. Thus, the agent responds to these dynamics to meet the goal. Agents also interact with each other either directly by communication or indirectly from changes of the common environment. Agents may need to exchange their view and their knowledge to reach a global objective. To this end, agents communicate to request and deliver information.

The dynamics of the environment depends on individual actions and also on the joint actions. By coordinating their action, the joint action of agents can fulfil the global objective. Therefore, it is essential to determine the interactions between agents.

The congestion control problem is decomposed into a series of single-hop flow control tasks. Figure 3.2 depicts the control agent and the communication components in the link flow control model (LFC).

A control agent is present in each sensor and communicates with other agents. The network state monitor provides the information assessing congestion levels. The agent observes through the monitor and carries out actions through the packet handler. The perceived environment encompasses the buffer, the channel and received traffic. At relevant events, the packet handler requests a decision from the agent. The actions are the different transmission mode and the admission decisions. The next section presents these actions and the state of environment.

3.3 Actions of the Packet Handler

The objective of the control agent is to adjust either the transmission rate μ_i or packet generation rate r_i.

The flow regulations are performed at different layers. The first adjusts the local contention at the MAC layer, thus affecting the transmission rate. The second controls the admission of traffic from the application, influencing the packet generation rate. Figure 3.3 depicts the communication layers and these flows.

In the methods proposed in the next chapters, the packet handler carries out actions modifying either the contention or the rate at which new packets are generated. Rate regulation is more effective than contention regulation to mitigate congestion. However, it is only possible with a periodic workload. Controlling the rate of random event has no sense. In Chapter 6, one regulation is assesed at the same time.

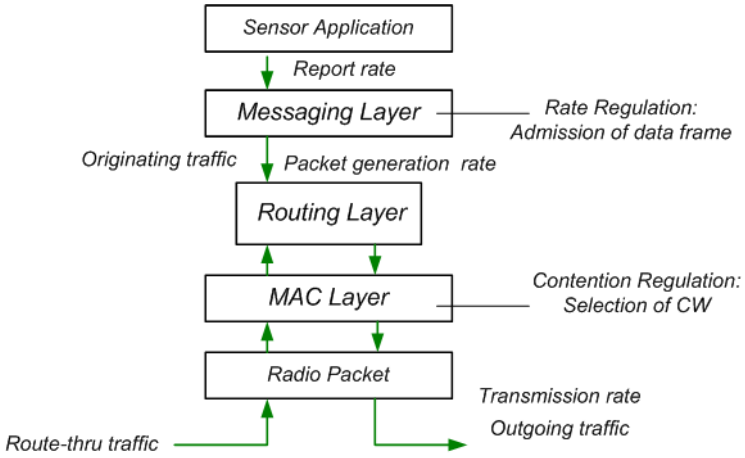

Figure 3.3: Regulation of the packet generation rate and transmission rate

3.3.1 Contention Regulation

Congestion at the transport layer often has its origin at lower layers. Buffer overflows can be prevented by lowering the arrival rate or by raising the transmission rate. Unfortunately, controlling accurately these rates is difficult when the channel usage is high. The *achievable transmission rate* depends on the local contention level i.e. the opportunism of the node. To prevent collision, random deferment intervals are used in contention based communications. The maximum deferment or contention window (CW) differentiate channel access among contending nodes. Figure 3.4 illustrates the decision made by the agent in the selection of the contention window and its effect on the transmission rate.

The transmission rate $\mu_i^{t_1}$ is simplified to a monotone function of the contention window size i.e. $\mu_i \simeq \Omega'(CW_i)$. Congestion are alleviated by controlling the access on the shared channel. The contention policy is approximately as follow:

1. When the local buffer is congested, the node is granted a privileged access by shortening the contention window.

2. When the parent node is congested, a node increases the contention window to

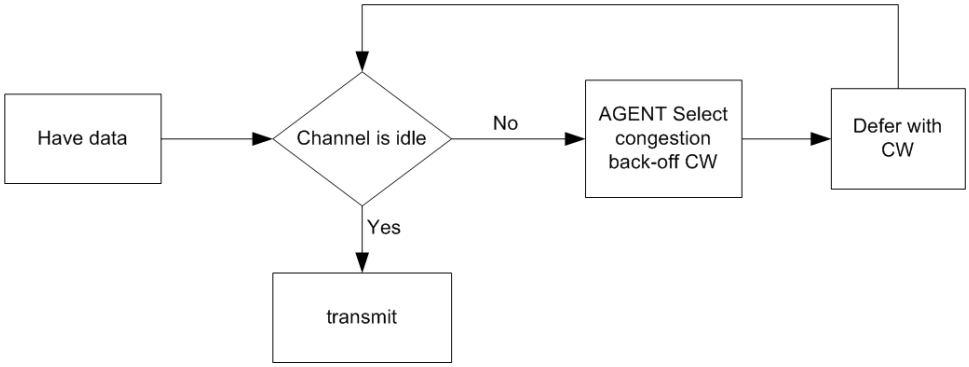

Figure 3.4: Flowchart of CSMA with back-offs

defer the transmission, and avoid overflows.

It is noted that when the two conditions are true, the node defers the transmission (2 overwrite 1) since the packets of the parent node are more "valuable".

Regulating the local channel contention helps to control transient congestions within a contention domain. Packet drops are reduced by distributing the packet accumulated in the buffers over several nodes. Contention regulation reduces but does not completely remove packet drops. When the offered load permanently exceeds the channel capacity, a different strategy consists in reducing directly the packet generation rate. Contention regulation is the solution to transient and local congestion.

3.3.2 Rate Regulation

Reducing the *packet generation rate* leads to an immediate reduction of the total traffic. The delivery of a single packet costs to the network a number of forwards. If this packet is not admitted in the network, bandwidth is made available for more than one transmission. Although an overload is quickly reduced by cutting the rate of the distant sources, the network would suffer from unfairness.

Without a fair rate control, some sources have a high report rate compared to others which experience bandwidth starvation. Packets originating near sink tend to be

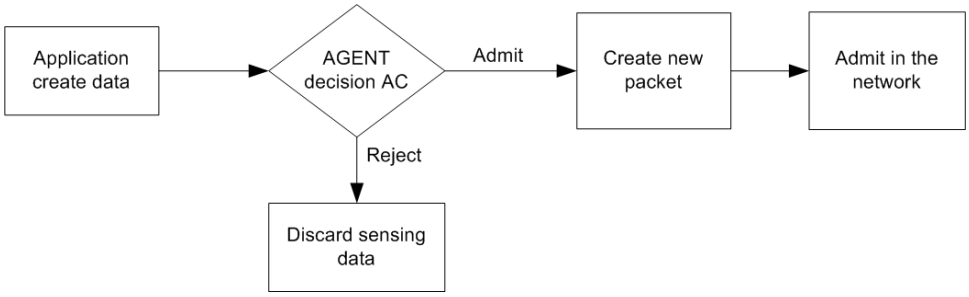

Figure 3.5: Flowchart of a packet generation

delivered in larger proportion than those from deeper in the network. In event based reporting, the flows are not continuous thus it is difficult to compare their rates.

The rate regulation is on a packet basis rather than on a rate basis because a given rate is hardly achievable with congested channels. The rate regulation is performed on the originating traffic. Figure 3.5 illustrates the agent deciding the admission of the sensing data into the network. Another option is regulating the route-thru traffic, but this requires informing sources at several hops away. The decisions $AC_i = \{ADMIT, REJECT\}$ affect the generation of originating packets. The expected decision determines the packet generation rate r_i.

The rate regulation prevents the load from exceeding the network capacity. An ideal scheme allows fair delivery without loss. The network is enforced to be fair at any load so that congestion may not develop. This technique is inspired from an exact rate adjustment in the work of Tien Ee [6]. Specifically, the available bandwidth B at one node is divided evenly among the N other sources in the upstream subtree. The packet generation rate r_i also must be smaller than the rate of the parent node r_p.

$$r_i = \frac{B}{N} \, and \, r_i \leq r_p \tag{3.1}$$

3.4 Network State Monitor

The state of a sensor node indicates the level of congestion and unfairness with local information. Four measurements are made: the buffer occupancy, the route-thru traffic rate, the upstream fairness index and the downstream fairness index.

Instantaneous buffer occupancy

The buffer occupancy is simple to obtain and makes a reasonable indicator of local congestion. In multi-hop communications, the state of the link indicates how well the packets are forwarded. Since a link is considered as congested if the next hop buffer is full, the buffer occupancy of the next hop node is an indicator of the link congestion.

Route-thru traffic rate

The traffic to forward, λ, measures the risk in terms of packet drops. Large traffic intuitively causes in case of congestion more drops than smaller traffic. And a node is more vulnerable to buffer overflows with more load. Congestion is frequently observed at proximity of the sink where nodes forward a large amount of traffic. The average traffic rate is the inverse of the average inter arrival time. The packet arrivals intervals i^t are averaged with the exponential moving average (EWMA):

$$\lambda^t = \frac{1}{i^t_{avg}} \tag{3.2}$$

$$i^{t+1}_{avg} = (1 - w_{tr}) \times i^t_{avg} + w_{tr} \times i^t \tag{3.3}$$

Upstream Fairness Index

Congestion causes packet loss and subsequently unfairness in a forwarding network. Although rate control can make a network efficient, it can be a source of unfairness if

not performed collaboratively. A metering technique is introduced to detect unfairness between sources.

To obtain a fair allocation of the bandwidth, the available bandwidth B at one node is divided evenly among the N sources in the upstream subtree. A fair packet generation rate r_i satisfies the equality [6]:

$$r_i = \frac{B}{N} \tag{3.4}$$

In order to accommodate burst packets, a token bucket is used like with Fusion scheme [4]. The sensor nodes determine their value of N by keeping a record of the origin of the packets.

Each intermediate node performs a metering of the route-thru traffic with a token bucket. One token is added for every N packets forwarded, until it reaches a maximum. This way, the agent does not need to determine the available bandwidth B but rather the outgoing bandwidth utilized. A counting of packets sent is sufficient to obtain B. A packet generated locally consumes one token. The state component is the current number of tokens $TokenT$.

$$0 \leq TokenT \leq TOKEN\_MAX$$

Downstream Fairness Index

A second index measures short term unfairness between the node and its parent. The rate of a source is determined with the smallest rate on the path from the source to the sink (c.f. traffic shaping in Chapter 2). The maximum rate can only decrease as the distance from the sink augments. Thus, the generation rate of a source r_i can not exceed the generation rate of the parent r_p.

$$r_i \leq r_p \tag{3.5}$$

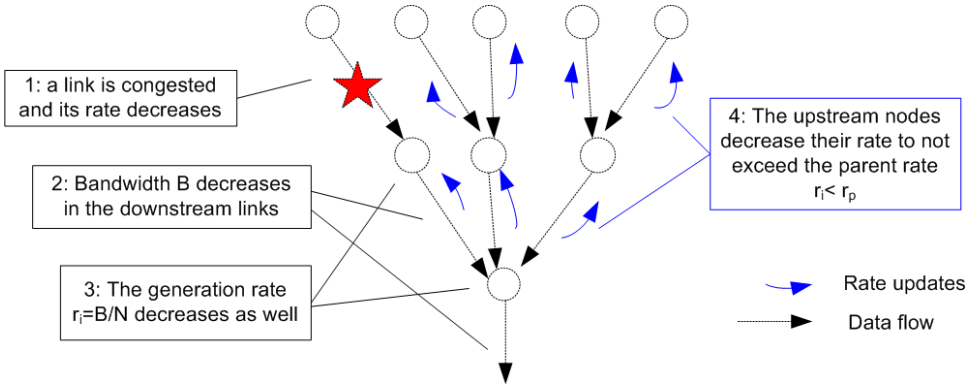

Figure 3.6: Rate limitation with fairness indexes

The second bucket collects one token every time it hears one packet generated and sent from the parent node. Similarly, the generation of a packet removes one token. There are currently $TokenP$ token in this bucket.

$$0 \leq TokenP \leq TOKEN\_MAX$$

With both rate limitation, congestion in any branches of the network will cause a decrease throughout all others parts of the network as illustrated in Figure 3.6.

A fair rate is under constraint from the upstream and downstream nodes. An unilateral rate limitation is insufficient to force a sensor node to share the same rate as its upstream subtree. The two constraints are verified if packets are generated only when *both buckets* are not empty. Since the sensor cannot generate new data when the bucket is empty, r_i is kept inferior to r_p. For one packet generated, N packets are expected to be forwarded, and one packet from the source parent is expected.

3.5 Conclusion

The control agent was presented with its observed environment and its actions. The use of buffer occupancy, the arrival rates and the rate control are widespread in con-

gestion control literature. The contention regulation and the fairness indexes are our contributions.

To solve efficiently the congestion problem, two solutions to the multi-agent problem are explored. In Chapter 4, a method based on reinforcement learning is suggested to adapt the contention and rate strategy to the topology. In Chapter 5, an inference method is used to coordinate actions and reduce congestion.

Chapter 4

Adapting Policies by Reinforcement Learning

In this Chapter, a solution method based on reinforcement learning is introduced. First, the semi-markov model and the solution approach are motivated. Before presenting the particular method used, an introduction to reinforcement learning is given. The second section describes the implementation of the algorithm on the congestion control problem. For both contention and rate regulation problem, the state and reward are defined.

4.1 Introduction

Sequential decision making in uncertainty motivates a Markov decision process model [13]. The environment of the agent is stochastic because the medium is accessed randomly. For example, the queue length at the next decision epoch is conditioned by a probability distribution.

In practice, the probability structure of the underlying Markov chain is difficult to obtain. The determination of transition probability is prohibitive and unrealistic as each node has its own perception of the environment. In addition, the topology is

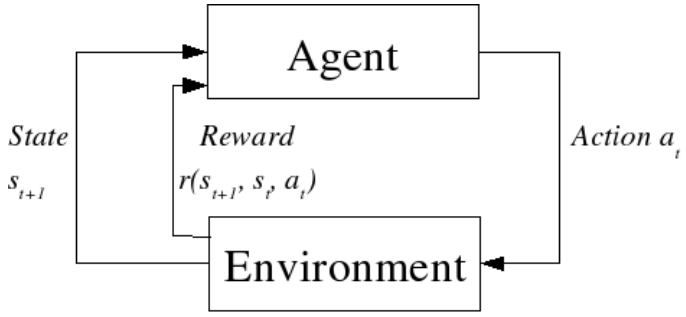

Figure 4.1: Reinforcement learning model

inherently spontaneous and thus it is impossible to predict the spanning tree and the traffic patterns.

This thesis investigates the learning of the policy from experience. With on-line learning, the sensor node can adapt by itself to the environment presented. A learning method provides more flexibility and robustness compared to a fixed strategy. The goal of the learning agent is to avoid congestion. It can be achieved by limiting the packet drops or unfairness. Contention and generation rate policy are learnt to provide coordinated actions.

4.2 Background on Reinforcement learning

Reinforcement learning (RL) [13] is a computational approach for goal oriented decision making. RL is a way to teach agent the optimal control policy. It is based on stochastic approximation through direct interactions with the environment. The decision maker is assigned rewards or punishments based on the temporal feedback of its action. The computational complexity of RL algorithms is considerably less than methods of dynamic programming since transition and reward matrices are not required.

Any reinforcement learning system basically comprises an agent, the environment, a reward function, a policy and a value function. The agent selects a control action a which leads to a transition from the environment state s. Upon the next decision,

the agent updates the new state s' from the sensory inputs and collects a feedback in the form of reward. The policy is learnt progressively at each iteration as illustrated in Figure 4.1.

The reward $r(s, a)$ defines the goal. It maps each state-action pair to a number that measures the immediate desirability of the action. A reward may be negative if the action results in unwanted consequences. RL problems are credit assignment problems because the immediate reward can be followed by worse ones. The objective of the agent is then to maximize the reward or credit in the long term. An optimal sequential decision path maximizes the expected reward and can be represented in more compact form with a *policy* π. The policy is an association from the context or state to the action to perform. Value function defines the desirability of a state (or a state-action pair) in the long term. It represents the expected average reward by following the policy starting from that state. Since the policy is based on value functions, an optimal policy is approximated completely when all the states and actions are visited a large number of times. Exploratory actions are taken for the purpose of visiting new states. The learning phase typically ends when the value functions have converged.

The computational burden is only related to the approximation of the values functions. Sutton [13] developed incremental learning techniques for control also referred as temporal difference methods. For an in-depth presentation of the solutions of reinforcement learning, the book by Sutton and Barto [13] is a reference in this topic.

4.2.1 SMART reinforcement learning

Sequential decision problems that are amenable to the Markov analysis are Markov Decision Processes (MDP). In a Semi Markov Decision Process (SMDP) [14], decisions are not restricted to discrete time epochs like in MDPs. Decisions are made whenever the system enters a specific state that is relevant to the agent. The state may change

several times between two decision intervals since the environment is a renewal process.

Most of research and applications focus on discounted sum of rewards as the optimality metric. The future rewards have less importance than the present reward with the discount criterion. However, in engineering problems, performances are preferably evaluated with their time average expected reward. Some examples are average efficiency or average throughput.

Let $R^*(s,a)$ be the expected average adjusted value by choosing actions optimally starting from the state s and action a. The value of state s can be written $R^*(s) = \max_a R^*(s,a)$. The interaction a_t at time t brings the environment from a SMDP state s to another SMDP state s' with the state transition probability $P_{s_t,s_{t+1}}(a_t)$ and sojourn time $\tau(s,a)$. There exists a scalar ρ^* satisfying the Bellman optimality equation in SMDP states:

$$R^*(s,a) = \max_a \left[r(s,a) - \rho^* \tau(s,a) + \sum_{s'} P_{s,s'}(a) R^*(s',a) \right] \tag{4.1}$$

An incremental learning procedure known as Semi-Markov Average Reward Technique (SMART) [14] allows the agent to accumulate knowledge on the states, immediate reward, and time spent between state-transition.

The SMART algorithm approximates the values $R(s,a)$ with the temporal differences method which assigns utility on one step performance (Eq. 4.2). If this error is positive, then the preference for the previous action on the previous state is intensified. Otherwise the tendency to take this action on that particular state is decreased.

$$\Delta R = r(s_t,a_t) - \rho_t \cdot \tau_t + \max_{a_{t+1} \in A} R_{old}(s_{t+1},a_{t+1}) - R_{old}(s_t,a_t) \tag{4.2}$$

$$R_{new}(s_t,a_t) = R_{old}(s_t,a_t) + \alpha_t \cdot \Delta R \tag{4.3}$$

where α_t is the learning rate parameter at the t^{th} decision epoch.

The average reward rate ρ at the t^{th} decision epoch is estimated by taking the ratio of the total reward earned and the total simulation time.

$$\rho_t = \frac{\sum_{k=1}^{t} r(s_k, a_k)}{\sum_{k=1}^{t} \tau(s_k, a_k)} \tag{4.4}$$

The control policy π is obtained by selecting the action with the highest the state-action value (for a maximization).

$$\pi(s_t) = a = \arg \max_{b \in A(s)} R(s_t, b) \tag{4.5}$$

An action different than the greedy action is chosen with probability ϵ_t. These exploratory actions ensure that all state-action pairs of the underlying Markov chain are visited and have been assigned a value.

An algorithm of SMART is given in Appendix.

4.2.2 Distributed reinforcement learning in cooperative systems

Reinforcement learning methods mostly focus on control problems in single agent settings. With multiple agents, the concatenation of individual states gives an exponentially large system state. To address control problem in a multi-agent setting, several distributed reinforcement learning methods were proposed. A taxonomy of multi-agent reinforcement learning is presented in the survey of Veloso and Stone [15].

Schneider at al. [16] proposed several heuristic methods based on the combination of values functions or rewards from several agents. The collaboration degree with a neighbor is determined with a weighting function. Since messages are exchanged at each steps of the learning, delay in communication can lead to a slow learning or non-convergence.

Guestrin et al. proposed a *planning* method for multi-agent factored MDP [17]. The

structured representation of the system allows the expression of the value functions in linear combination of basis functions. Since a basis function relates to only a small number of agents, the complexity is reduced exponentially. Finally, the weights of each basis function are computed with a linear program. The computations are carried out off-line and centrally.

In [18], the same author introduced a collaborative action selection method. First, the global payoff is approximated with a sum of local payoff functions depending on at most two agents. Since an agent can affect only a few others, the multi-agent system can be represented with a *coordination graph*. An edge symbolizes an interdependency between two agents. Next, the agents run the *variable elimination* algorithm which finds the exact solution to the maximization problem. This distributed action selection strategy is the key idea of other multi-agent extension of RL methods such as Q-learning.

However, the variable elimination algorithm requires the synchronization in the elimination process. The decision latency increases with the size of the network because of synchronization. The above methods may not be suitable to the congestion problem since real-time response is critical to avoid congestion in WSNs,

4.3 Reinforcement Learning for congestion control

Since the wireless sensor nodes are geographically scattered and have limited communication range, the learning has to be decentralized in order to be communication efficient. If two nodes are apart of several hops, their coordination would be expensive in communication. Consequently, the scope of the coordination is limited to a local neighborhood.

The congestion control problem is decomposed into several overlapping link flow control (LFC) problems. The sub-problems are interrelated, but solved independently at each node. One learning agent is set at each sensor node. A Markov decision process characterizes this agent and the solution is approximated by reinforcement learning.

With relevant state and reward, the local optimizations may imply a global near-optimal system. However, in the objective of enabling coordination, the interactions related to congestion are to be considered.

The SMART algorithm operates in continuous time and does not need synchronization. In multi-agent settings, one agent should not synchronize with another, so that the system is robust to failures. This is especially true when channel contention delays the transmission randomly. Consequently, the information exchanged between agents are processed asynchronously. The coordination messages are incorporated as they are received.

The network state monitor is updated continually and the reward is cumulated until the next action. Although the agent may update the state more than once between two consecutive actions, the decision is made normally according the current policy and action selection strategy.

In the following, the state and rewards are established for the congestion control problem. As mentioned in the previous chapter, the transmission rate can be controlled to mitigate congestion. This control is achieved by regulating the contention window when the channel is highly loaded. Next, the control of the packet generation rate is formulated. The objective is then to reduce unfairness by limiting the rate.

4.3.1 Reinforcement Learning of Contention Window Policy (RLCW)

The contention window determines the probability of transmission and thus the transmission rate. Packet drops from congestion are located at the buffer level. The concept of RLCW is to learn the policy which prevents packet drops at the *next hop node*. The common approach [19] is to learn the policy which minimizes *local* drops. Given the multi-hop nature of communications, a sensor node has to evaluate how well its packets are forwarded. Therefore, a state of the link is better defined as the state of buffer at

the next hop node.

Sensor nodes consider the current state of buffer to select the back-off decision. The value functions learned estimate future rewards. As a result, the reinforcement learning provides a control policy that *anticipates* the buffer lengths and the buffer overflows. With the average expected reward criterion, the policy will minimize the long term average drops on the link.

State and action space

The state descriptor for a single agent is the queue length of the parent node q_p.

$$s = [q_p]$$

Before each packet transmission attempt, the control agent updates the state s, collects the reward r and then takes an action a. The action is the choice of the contention window CW from a finite set.

$$a \in \{CW = c_0, CW = c_1, \ldots\}$$

The feedback signal is the cost related to drops on both sides of the link. The packet drops are the cause of inefficiency, unfairness and low application reliability. The Distributed Reward RL heuristic [16] is used to learn coordinated actions. The reward is the weighted sum of two costs. D_l denotes the local drops between two transmission attempts. D_p denotes the drops at the parent node during the same period.

$$Reward = -Cost = -(w_l D_l + w_p D_p) \tag{4.6}$$

This cost is equivalent to a negative reward. Since packets dropped with higher hop

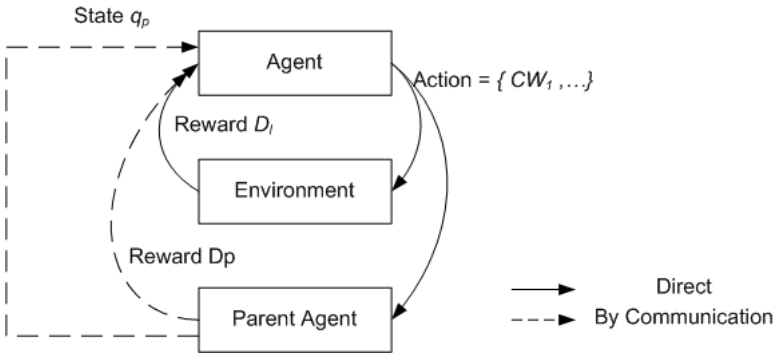

Figure 4.2: Agents and Communications involved in RLCW.

count cost more energy, a parent drops weights $w_p = 1.0$, more than local drops with $w_l = 0.01$.

$$r(s_t, a_t) = -\tau(s_t, a_t) \times Cost$$

The interactions between agents are summarized in Figure 4.2. The agent controls the contention level of the node through its direct action. If the action was notified to another agent, all nodes must have same set of actions limiting the application to homogenous network. Nodes advertise their current buffer length and drops. This contrasts with the backpressure method [4, 3] where congestion is detected locally but stopped from upstream.

The control agents are able to coordinate by mutual observation. The state is actually *non-local* because the agent "observes" the environment of the parent node. The reinforcement signal is the weighted sum of rewards from several agents. This method is known as *distributed reward RL* [16] and converges to better solutions than independent Q-learning in cooperative scenarios. The learning agent can choose the best alternative if the choices all lead to congestion. For example, if a node and its parent have their buffers congested, packet drops are likely to occur in both nodes. A short back-off drains the local buffer more quickly and prevents local overflows. However, this can lead to

the overflow of the parent node. With distributed reward RL, the agent considers the feedback of both agents. After repeated trials, the agent can estimate the action with the least worse outcome since rewards are real valued.

A pseudo code of the algorithm is provided in Algorithm 1.

Pseudo code of the algorithm in RLCW

```
Initialize SMART learner
for each packet transmission attempt do
    Collect D_l since last attempt;
    Collect packet drop advertised by parent node D_p since last attempt;
    Set Cost = (w_l D_l + w_p D_p) ;
    Determine the last update of parent node buffer q_p;
    Set the state S = [q_p];
    Update the SMART learner ;
    Obtain the action a from the SMART learner;
    Advertise D_sum and q_p in the header; Set the back-off window according to a;
    Start back-off timer;
    Upon timer expiration:
    if channel idle then
        Transmit ;
        D_l = D_p = D_sum = 0;
    else
        Wait for channel idle ;
        D_sum = D_sum + D_l;
    end
end
```

Algorithm 1: RLCW Algorithm: Learning of the contention policy

4.3.2 Reinforcement Learning of Rate Policy (RLRATE)

Rate regulation in a sensor network should be a collaborative task if fairness is design criterion. A sensor network may not drop packet, but the packets collected from each source present large disparities. The regulation of the rate has two objectives: prevention of congestion and fair allocation of the available bandwidth. The states are indicators of fairness. They indicate whether the local source has generated more packets in comparison to its descendent and parent nodes in the spanning tree.

State and action space

The state descriptor of the RLRATE agent is:

$$s = [TokenT, TokenP]$$

$TokenT$ is an indicator of the fairness between the packet generation rate and the route-thru traffic rate. $TokenP$ indicates the difference of the local packet generation rate with the parent node's rate. $TokenP$ is a shared state because a node and its parent directly affect it as illustrated on Figure 4.3.

The sensing data is either admitted or not in the network by generating a new packet. The average action taken determines the packet generation rate (originating traffic rate). For instance, if the frames are always admitted, the packet generation rate is equal to the sensing rate. The rejection of frames intends to reduce the traffic moving in the network, and subsequently to limit unfairness.

$$a \in \{ADMIT, REJECT\}$$

The objective of the agent is keep the network fair by maintaining the following

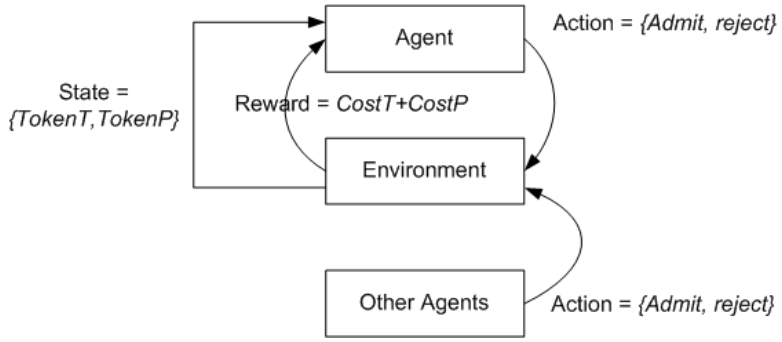

Figure 4.3: Agents and Communications involved in RLRATE.

conditions discussed in Chapter 3:

$$r_i \leq \frac{B}{N} \text{ and } r_i \leq r_p$$

The actions which go against these conditions are punished with negative rewards (Eq. 4.7 and 4.8).

The reward is negative if the sensor node generates a new packet although the node was seen greedy. An empty token bucket indicates that the originating traffic has exceeded the *fair share* of bandwidth.

$$Cost = CostT + CostP$$

$$CostT = \begin{cases} 1 & \text{if } TokenT = 0 \text{ and a=ADMIT ,} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

$$CostP = \begin{cases} 1 & \text{if } TokenP = 0 \text{ and a=ADMIT ,} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

A pseudo code of the RLRATE method is given in Algorithm 2. The interactions between agents are illustrated Figure 4.3. The state and reward are updated locally without transmission involved. As a result, the contention delay has no effect on the performance.

Pseudo code of the algorithm in RLRATE

```
1  Initialize SMART learner
2  for each report do
3      Determine the number of tokens tokenT and tokenP;
4      Set the state s = [tokenT, tokenP] ;
5      Calculate CostT, CostP with the current state and lastAction;
6      Set Cost = CostT + CostP ;
7      Update the SMART learner;
8      Obtain the action a from the SMART learner;
9      if (a = ADMIT) then
10         Enqueue and Send the report;
11     else
12         Discard the current sensing report;
13     end
14     Record lastAction = a;
15 end
```

Algorithm 2: RLRATE Algorithm: Learning of the generation policy

4.3.3 Implementation issues

Inclusion of the exploration phase in the evaluation

The first phase of learning has exploratory actions, thus performances is not representative of the learned policy. With the appropriate exploration rate decay, the length of the exploration phase can be adjusted. A decaying exploration rate improves the learning process in the beginning. However, for a fair comparison with other schemes, the early results from the exploration should be taken in account. This exploration phase becomes negligible as the duration of evaluation increases.

Processing and communication overhead

The processing in one step of learning only needs addition, multiplication and a random number generator. With parameterized values functions, the processing complexity is only linear in the number of actions and of state components.

With RLCW method, the buffer length and drop rate are communicated to neighbors. To avoid communication overhead, normal data packets piggy back these control information (two integer numbers). With RLRATE method, the states are obtained locally and agents do not communicate.

Function approximations

Real world RL problems are intractable as the state space grows exponentially with the number of state variable. The learning is slow if the state space is too large. Visiting all the states a sufficient number of times may pose a problem. Therefore, function approximations that can generalize state features are essential to accelerate learning. The memory used for storage is reduced by sharing what was learnt on one state with similar features.

A function approximator called Cerebellar Model Articulation Controller (CMAC) is frequently used in reinforcement learning [20]. It is linear unlike neural networks. A CMAC partitions the input space into overlapping tiles. Each tile ψ or parameter captures a salient feature in the state descriptor and performs a local generalization. The value of a state becomes a linear combination of weights w.

$$V(s, w) = \sum_{k=1}^{K} w(k) \cdot \psi(k)$$

The number of parameters K is far less than the number of states.

In the learning of the value, the TD update (Eq. 4.3) is replaced by a step which updates the weights according to the back propagation algorithm. The reinforcement is shared between the parameters $i \in \mathbf{P}(s)$ present in the current state.

$$w_i^{t+1} = w_i^t + \alpha \Delta w, \text{ for } i \in \mathbf{P}(s)$$

$$\Delta w = \frac{\Delta R}{card(P(s))}$$

Each tile is linked to a single state to avoid exponentially many tiles. The value is a linear combination of functions that depend on just one state component. This simplification removes the possibility of learning features combining several state components.

4.4 Conclusion

This chapter presents a learning approach to address the congestion control using states and rewards that are linked to other agents to enforce coordination.

With learning based methods, RLCW and RLRATE, the policy is constructed step by step according to a feedback. In RLCW, the contention windows are controlled with a policy obtained by reinforcement learning. Packet drops are the feedback signals and

to be minimized. In RLRATE, unfairness is detected locally and the agent learns to bring balance by controlling the generation of packet.

The following chapter explores the use of an inference algorithm to solve the contention regulation and rate regulation problem; this algorithm exploits a priori known interactions rather than approximating them from experience.

Table 4.1: Summary of the learning methods

Method	Action	State	Communication
RLCW	Contention Window	Buffer length	Buffer length q_p and Drop count D_p
RLRATE	Admission/ Rejection of packet	Token bucket contents	None

Chapter 5

Distributed Coordination using Inference

In this chapter, an alternative method to reinforcement learning is explored. The congestion control problem is modelled as a coordination problem. An iterative algorithm related to belief propagation is explored as a coordination method. A background is given in the first section, then the contention and rate regulation problem are modelled.

5.1 Introduction

Coordination is the agreement on actions in a control problem involving multiple actors. Some MAC or transport protocols are examples of coordination in WSNs. Inefficiencies in a sensor network can be the result of buffer overflows. The overflow of the downstream buffer can be avoided by reducing the transmission rates of upstream nodes. Therefore, flow control is a coordination problem between congested nodes and their neighbors.

A WSN is evaluated as a whole system, since sensing is a collective task. In a

distributed system, local optimizations often preclude global optimality. Local optimal decisions can be conflicting with each other. Since packets are broadcasted, a local congestion solution can worsen the channel conditions in different part of the network.

This global optimization problem can be formulated as an inference problem in a Bayesian network. Given the state of sensors $\vec{s}$, the problem consists in finding the joint action $\vec{a}$ such that the joint probability distribution of congestion is minimized network wide.

$$\vec{a} = \arg\min_{\vec{a}} Prob(\text{congestion}|\vec{s}, \vec{a})$$

With the computation and communication capabilities, wireless sensors can benefit from the power of distributed algorithms. An exhaustive search is costly as there are exponentially many action vectors. A centralized computation is unrealistic in WSNs because of the induced communication delay and overhead. Moreover, real-time decision making is expected as buffers are small and can overflows within a second.

To this end, a complex control problem is formulated as a minimization problem of reduced complexity using the inference method. All the agents take decision in parallel and exchange messages to make sure local decisions lead to good team behavior. The optimal action of the agents is the unknown in the problem.

5.2 Belief Propagation

Belief Propagation is an algorithm for solving probabilistic inference problems in Bayesian networks [21]. Its application generalizes to undirected graphical models or Markov random fields (MRF). Figure 5.1 illustrates a MRF as a graph $G = (\mathbf{V}, \mathbf{E})$. A vertex $v \in \mathbf{V}$ is associated with a random variable $X_i \in \mathbf{X}$. A *local potential* $\psi_i(X_i)$ depends on this variable. An edge $e \in \mathbf{E}$ represents a dependence between two variables, which is defined with a *compatibility potential function* $\psi_{ij}(X_i, X_j)$. Considering that all the variables

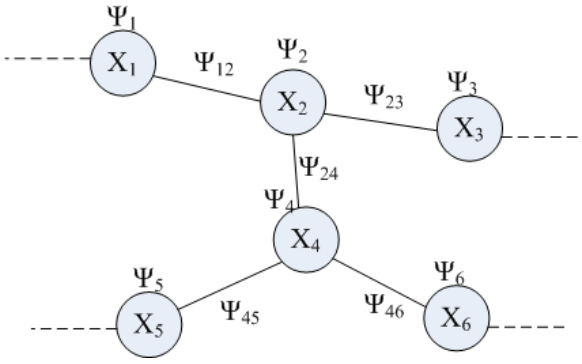

Figure 5.1: An undirected graphical model with the potential functions

dependencies are expressed with the functions ψ_{ij}, the joint probability function is the product of distributions over cliques:

$$Prob(\vec{X}) = \prod_{i \in V} \psi_i(X_i) \prod_{(i,j) \in E} \psi_{ij}(X_i, X_j) \tag{5.1}$$

A common query in a MRF is the Maximum a Posteriori (MAP) configuration of variables:

$$\vec{X}^* = \arg\max_{\vec{X}} Prob(\vec{X}) \tag{5.2}$$

Belief propagation is an iterative method to solve this maximization problem. In the algorithm, each node i sends a message m_{ij} to any neighbor j in the graph. $N(i)$ denotes the set of neighbors of node i in the graph.

$$m_{ij}^{(t+1)}(X_j) = \max_{X_i} \left\{ \psi_i(X_i)\psi_{ij}(X_i, X_j) \prod_{k \in N(i) \setminus j} m_{ki}^{(t)}(X_i) \right\} \tag{5.3}$$

After reception of new messages, the belief is updated and new messages are sent to neighbors based on what was received previously. Since the messages propagates and are updated from a node to another, local information is aggregated and spread in the whole graph.

The algorithm converges in finite number of steps to the optimal joint action [21]. A fixed point is reached when the messages converge in value. The global optimal configuration is given by choosing locally X_i^* as in Eq. 5.4.

$$X_i^* = \arg\max_{X_i} \left\{ \psi_i(X_i) \prod_{k \in N(i)} m_{ki}^{(t)}(X_i) \right\} \tag{5.4}$$

The Max-Plus and Min-Sum algorithms belong to the family of belief propagation algorithm [22]. They employ any positive potentials functions rather than probabilistic distributions. J. Kok [23] studied Max-Plus to coordinate actions in multiple agent settings. The coordination dependency between two agents is represented with a joint payoff function.

The *Min-Sum* algorithm is similar to Max-Plus, but minimizes a global value rather than maximizes it.The global cost is decomposed in simpler terms depending on actions a:

$$Cost(\vec{a}) = \sum_{i \in V} \psi_i(a_i) + \sum_{(i,j) \in E} \psi_{ij}(a_i, a_j)$$

The minimum cost action in a coordination graph is equivalent to the maximum a posteriori configuration in an undirected graphical model. The optimal joint action a_i^* is found with similar messages passing (Eq. 5.5). These messages are actually estimates of the global cost function from a single agent's perspective. Node i gives a value m_{ij} for each action a_j of j according to the information available and collected so far.

$$m_{ij}^{(t+1)}(a_j) = \min_{a_i} \left\{ \psi_i(a_i) + \psi_{ij}(a_i, a_j) + \sum_{k \in N(i) \setminus j} m_{ki}^{(t)}(a_i) \right\} \tag{5.5}$$

The messages received locally determine the local action causing the least global cost. The joint optimal action is calculated distributively and in parallel at each node.

The following consistency property holds in graph without cycle after convergence:

$$\min_{\vec{a'}|a'_i=a_i} Cost(\vec{a'}) = \psi_i(a_i) + \sum_{k\in N(i)} m_{ki}^{(t)}(a_i) \tag{5.6}$$

Consequently, each agent i can select individually optimal action a_i^*.

$$a_i^* = \arg\min_{a_i} \left\{ \psi_i(a_i) + \sum_{k\in N(i)} m_{ki}^{(t)}(a_i) \right\}$$

The family of belief propagation algorithms illustrates the transformation of a complex optimization problem into a set of smaller problems. Each agent solves a problem with few variables and only uses local message passings. Convergence proofs and strong theoretical results exist for the belief propagation algorithm and its extensions [22]. Due to its asynchronous nature, the algorithms based on belief propagation are particularly suitable for real-time decision making. A pseudo code of the coordination algorithm is provided in Appendix.

5.3 Definition of Potential Functions

5.3.1 Coordination Graph

The congestion at a sensor node is caused by a few neighbors rather than all of them. Packet drops result from the inability of the receiver to accommodate sender's packets. Therefore, pair interactions can be constructed as illustrated in the Figure 5.2.

A framework for multi-agent coordination [17] exploits these local dependencies to represent a large coordination problem with a coordination graph $G = (V, E)$. The global cost $\Psi_{\vec{a}}$ is decomposed into a sum of local functions (5.7). Function ψ_i is the cost that is caused unilaterally by node i. Function ψ_{ij} describes the cost additionally

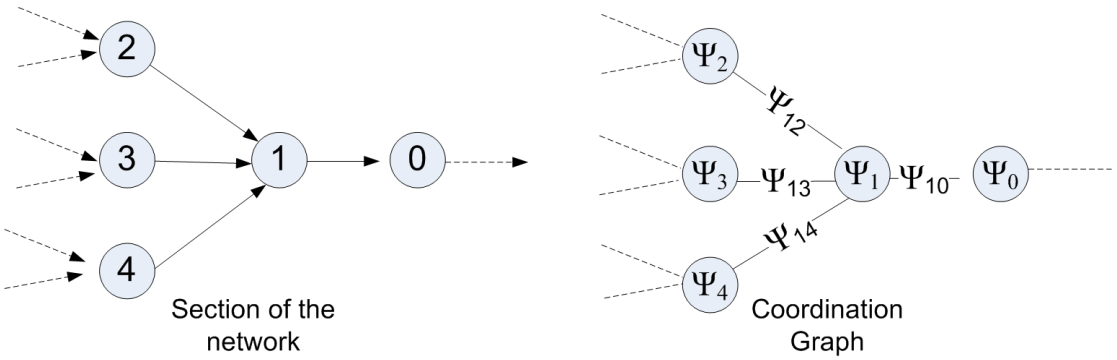

Figure 5.2: The coordination graph over the spanning tree: the links on the spanning tree correspond to the pairwise dependencies. The immediate neighbors in the routing tree coordinate their respective actions.

created when the neighbor nodes i and j choose respectively action a_i and a_j.

$$\Psi_{\vec{a}} = \sum_{i \in V} \psi_i(a_i) + \sum_{(i,j) \in E} \psi_{ij}(a_i, a_j) \tag{5.7}$$

The coordination algorithm minimizes a global cost with local computations and local communications. The Min-Sum algorithm is intended to minimize distributively congestion and unfairness.

5.3.2 Coordination of Contention Windows (COCW)

Contention regulation (c.f. Chapter 3) is the mean to adjust the transmission rate when the channel is congested. Potential functions express the congestion with the estimated packet drop. The variables in these functions are the contention windows CW. A small contention window increases contention and transmission rates. Consequently, it helps clearing the local buffer, but the packet accumulation is only transferred to the next hop node.

The potential functions in this study are based on intuition rather than rigorous analysis because the relationship between contention window and achievable rate is

difficult to determine distributively. The proposed heuristic method estimates the loose upper bound of packet drops.

The congestion likelihood increases with the number of upstream nodes. While child nodes increase the risk of congestion, each of then send information to the parent node. Consequently, the messages from child nodes add up (Eq. 5.6) to influence further the decision and to avoid mismatching actions.

Congestion potential

The total incoming traffic at node i is the sum of the route-thru traffic λ_i and the originating traffic r_i as illustrated in Figure 5.3. The transmission rate μ_i depends on the actions or contention window CW_i and has implications on the local congestion and on the downstream node j's congestion. For example, the starvation of an outgoing link ($\mu_i = 0$) is the worst case of buffer congestion. The packet drop rate is then nearly equal to the incoming traffic $D = r_i + \lambda_i$. A general trend can be drawn: the expected packet drops increases with traffic intensity.

The local potential $\psi_i(CW_i)$ is function of the local buffer and the contention window which are the state and action, respectively. A nearly full buffer is more likely to overflows than an empty buffer. Therefore, the congestion likelihood increases with the buffer occupancy q_i. We make the simplifying assumption that the transmission rate decreases linearly with the contention window. For example, doubling CW_i roughly halves the service rate and augments the buffer drops. The potential packet loss is given by ψ_i at Eq. 5.8 where CW_{min} is the shortest back-off window that always wins the contention (≥ 1).

$$\psi_i(CW_i) = r_i \frac{q_i}{q_{max}} (1 - \frac{CW_{min}}{CW_i}) \tag{5.8}$$

Taking decision independently leads to uncoordinated actions and congestion. The

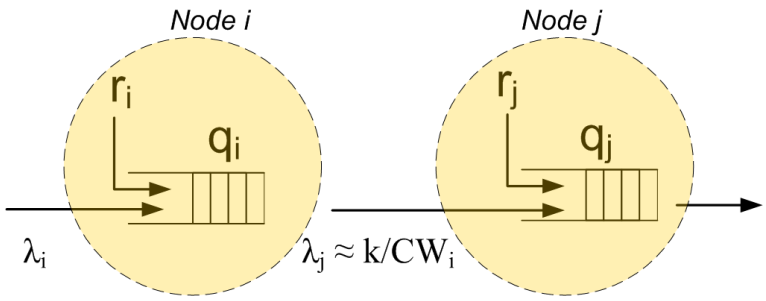

Figure 5.3: Coordination between two adjacent sensor nodes: the parameters taken in account are the packet generation rates r_i , arrival rates λ_i, queue length q_i, and contention windows CW_i

coordination function is defined as the congestion resulting from *unmatching* control actions: In Figure 5.3, if the node i uses a short contention window, j will send less packet and will receive more. A node which uses a short back-off confiscates the available bandwidth of the neighboring nodes. This node j is thus likely to have a congested buffer. The lack of coordination creates additional drops which are bounded by the traffic received at the congested node j. The buffer drops augment with the queue lengths q_i and q_j. On the one hand, the probability of buffer overflows at the downstream node increases with its occupancy q_j. On the other hand the length of the local queue q_i determines the congestion of the mentioned downstream node.

A coordination potential function ψ_{ij} is constructed from these observations:

$$\psi_{ij}(CW_i, CW_j) = 0.5\,\lambda_j \frac{q_i + q_j}{q_{max}} \cdot \frac{CW_j}{CW_i} \tag{5.9}$$

The function ψ_{ij} is a rough estimation of the packet loss for different contention windows. In a link with one upstream and one downstream node, the packet drops will occurs when the upstream node send more packets (with a smaller CW) than the downstream can receive. When $(2k \cdot CW_i) \leq CW_j$, $k \times \lambda_j$ packets can be be dropped because the upstream node i will win the contention more frequently at the expense of the downstream node j. When the contention level is equal $CW_i = CW_j$, half of λ_j can

be dropped. Actions are coordinated when the upstream nodes use a longer contention window than its downstream nodes. The congested node is able to drain its queue with high transmission rate.

These coordination functions are based on intuition, rather than rigorous analysis. With COCW, a sensor node advertises its current buffer length q_i, the incoming traffic λ_i and the messages m_{ij}. To reduce control packet overhead, this information is put on every data packets, and is collected by enabling overhearing. A pseudo code of COCW is given is Algorithm 3. In the followings, the rate control is modelled as coordination problem to minimize unfairness.

Pseudo code of COCW

1 Determine the incoming traffic λ_i and calculate the potentials functions
2 **for** *each packet to transmit* **do**
3 Determine the buffer occupancy
4 Select a neighbor j and compute the coordination message addressed to j
5 Collect messages and create function $g_i(a_i)$
6 Choose action a_i^* which minimizes the cost
7

$$a_i^* = \arg\min_{a_i} g_i(a_i)$$

8 Transmit with the contention window CW
9 **end**

Algorithm 3: Pseudo code of COCW

5.3.3 Coordination of packet generation rates (CORATE)

As presented in the LFC problem (c.f. Chapter 3), two rate constraints provide a fair and efficient rate:

$$r_i = \frac{B}{N} \text{ and } r_i \leq r_p$$

A possible decomposition of the total unfairness in the network into simpler terms is:

$$\Psi_{\vec{a}} = \sum_{i \in V} \psi_i(a_i) + \sum_{(i,j) \in E} \psi_{ij}(a_i, a_j) \tag{5.10}$$

Since the first constraint only depends on the local rate, the local potential $\psi_i(a_i)$ represents the first constraint into a cost function. The second equations involves directly the local rate and the parent node's rate, therefore, the coordination functions $\psi_{ij}(a_i, a_j)$ represents the cost of unmatching action pairs, specifically those decreasing fairness.

State of fairness

The state components in CORATE are $TokenT$ and $TokenP$. $TokenT$ is an indicator of the fairness between the originating traffic rate and the route-thru traffic i.e. the upstream sub-stree. $TokenP$ indicates the difference between the local packet generation rate and the parent node's generation rate. The node is considered *greedy* when $TokenT = 0$ or $TokenP = 0$.

Unfairness Potentials

The local potential $\psi_i(a_i)$ determines a cost when a packet is generated, although more originating packets than the fair share were already generated ($TokenT = 0$). The rejection of packet in normal conditions is also penalized to maximize throughput and reliability.

	$\psi_i(ADMIT)$	$\psi_i(REJECT)$
$TokenT = 0$	1	0
$TokenT > 0$	0	1

Each node maintains the $TokenP$ of its child nodes. By counting the packets originating from its child in the route-thru traffic, the parent node obtains this index. The parent node then sends the coordination message back to each one of the child nodes. This message is the preference of the parent for the actions of a particular child node. The decisions penalize *greedy* child nodes j ($TokenP = 0$) if they still generate packets at the next round. The message from the parent has more weight so that both constraints are verified together.

	$\psi_{ij}(ADMIT)$	$\psi_{ij}(REJECT)$
$TokenP = 0$ at j	2	0
$TokenP > 0$ at j	0	0

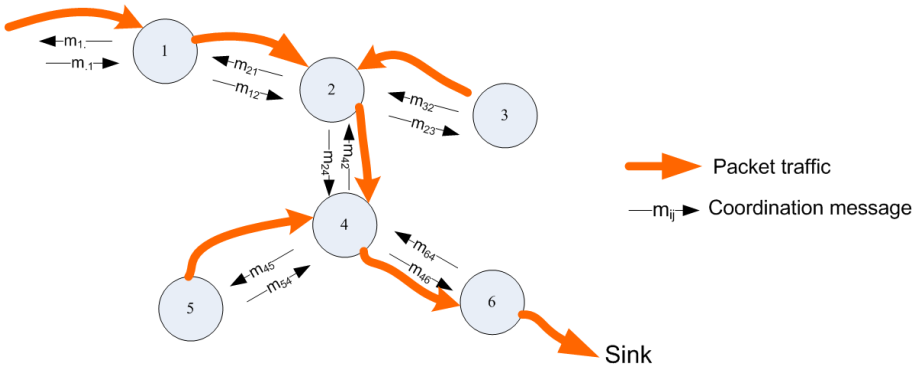

Figure 5.4: Implicit message passings with the primary traffic

Pseudo code of CORATE

Determine the buckets content $TokenT$, $TokenP$

Determine the $TokenP_j$ of each of the child node j

Update the potentials functions ψ_i and ψ_{ij}

for *each packet to generate* **do**

  Select a neighbor j and compute the coordination message addressed to j

  Create function $g_i(a_i)$ from the collected messages m_{ij}

  Choose action $a_i^* = AC_i$ which minimizes the cost

$$a_i^* = \arg\min_{a_i} g_i(a_i)$$

  Admit the packet for a_i^*=ADMIT.

end

Algorithm 4: Pseudo code of CORATE

5.3.4 Implementation issues

Message and computation overhead

A congestion control based on the Min-Sum algorithm can use the data traffic to carry the coordination messages (Figure 5.4). The use of specific messages packets creates too much overhead and has no benefit because two nodes cannot transmit simultaneously. Therefore, coordination message m_{ij} is *inserted within the header* whenever a packet is transmitted. Since a node has to pass a message to all of its neighbors, *the messages are addressed one by one in turn*. The destinations of these messages are the neighbors which have to *listen promiscuously to all* transmissions. The effect of this technique is evaluated in the simulations. The use of specific messages packets creates too much overhead and has no benefit because two nodes cannot transmit simultaneously.

Since the coordination graph is the spanning tree (has no cycle), the algorithm converges to the optimal solution in static cases. The messages values stabilize after the longest path experiences a forward and backward propagation of messages. The message overhead of the algorithm grows linearly with the size of the network. However, full propagation is *not necessary* as congestion spots are solved locally. This stems from the observation of Crick et al. [24].

For each decision, the processing complexity and storage requirement is linear in the number of neighbors N and in the number of actions. The computation is light for sensor motes since tens of additions and multiplications are carried out for each packet transmitted.

Correctness of decisions

Experimental results showed that asynchronous message passings do not affect the results. Asynchronous versions of belief propagation were experimented in distributed applications such as robot soccer [25, 24].

Belief propagation and its variants are robust to random communication errors. Crick and Pfeffer [24] conducted an empirical study on the effect of disturbances of the asynchronous algorithm. He concluded that unreliable communication affects little the correctness of beliefs [24].

The algorithm keeps the number of errors relatively low in highly dynamic states [24]. The current state determines the potential functions. In COCW and CORATE, the states are packet level measures which vary rapidly. Simulations will tell if the number of erroneous decisions is kept low.

J. Kok [23] mentions that the algorithm's performance declines when several optimal solutions exist. Agents tend to forget the good actions in the past and reach only sub-optimal configurations. However, the presence of outliers in the actions lessen this discrepancy. As such, the different control actions should have large differences to make the solution unique.

5.4 Conclusion

This chapter presents a new iterative approach to solving the congestion control using a decomposition of a global cost into local expression and a parallel message passing to select the action distributively. The key concept presented is the coordination of multiple agents' actions.

With the coordination based methods, COCW and CORATE, the interactions between agents are modelled explicitly unlike in the reinforcement learning approach. Heuristics were proposed to elaborate these potential functions.

The next chapter presents the simulations results of the proposed methods. Rate regulation methods, RLRATE and CORATE are evaluated on a periodic workload, whereas contention regulation methods RLCW and COCW are evaluated on an event

based workload.

Table 5.1: Summary of the coordination methods

Method	Action	State	Communication
COCW	Contention Window	Buffer length	Buffer length q_p and arrival rate λ + messages m_{ij}
CORATE	Admission/ Rejection of packet	Token bucket contents	messages m_{ij}

Chapter 6

Simulations and Results

In this chapter, the simulation results of the proposed methods are shown. The packet drops, throughput, energy efficiency and fairness are evaluated with random and periodic sensing scenarios.

We proposed learning methods (RLCW and RLRATE) and coordination methods (COCW and CORATE) to congestion control in WSNs. In the following sections, we define the simulation model and the evaluation methods. The work of Hull and Jamieson [4] is used as a baseline method in the comparison. The influences of the sensing rate on the performances are analyzed. In the first scenario, contention regulation methods are evaluated with a non-periodic workload. In the second scenario, the rate regulation methods are evaluated with a periodic workload. The period value is a supplementary information which gives to the second method better efficiency in handling congestion than the first method.

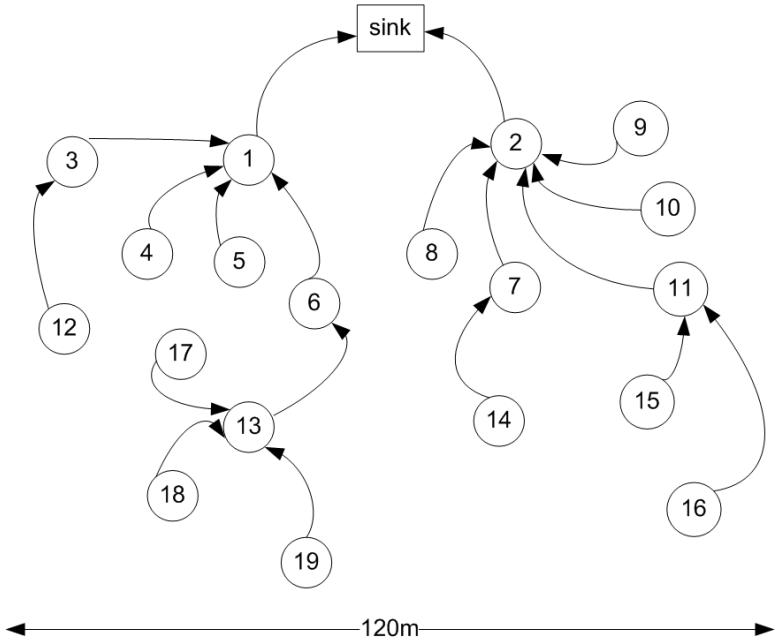

Figure 6.1: Topology of the simulated wireless sensor network.

Table 6.1: System parameters

Parameter	**Value**
Bandwidth	40 Kbps
Packet error rate	0.10
Data packet size	60 bytes
ACK size	13 bytes
Data rate	45 packets/s
Retransmission limit	7
Transmission range	45m
Carrier sensing range	150m
Propagation model	Two ray ground reflection
Buffer length	10 packets
Transmission Power	0.075 mW
Reception/Idle Power	0.025 mW
Contention window c_0	31ms
Contention window c_1	127ms
Token Bucket capacity	10

6.1 Model of wireless sensor network

6.1.1 Simulation parameters

The simulated wireless sensor network consists of one sink and 19 sensor nodes. The sensing area is 120x120m wide and the nodes are arranged as depicted in Figure 6.1. The optimal functioning point changes from a topology to another. Consequently, the results of several random topologies cannot be averaged. The experimented topology is sufficient to reveal the variation in depth within a routing tree. The packets are forwarded over multiple hops to the sink. The next hop node is manually chosen in the simulation. The paths to the sink are static and the topology as well.

Simulations were conducted on the network simulator NS2 [26] and carried out on a Pentium-M 1.8 Ghz. The code of IEEE 802.11 in NS2 was simplified to perform a simple CSMA: the back-off timer is not interrupted when other transmissions are sensed on the channel. This is to make the medium access in the simulation more similar to MAC for WSNs. The wireless environment is simulated with the two-ray ground reflection propagation model and an imperfect channel. The packet error rate is set to 0.10 and the nominal reception range is 45m. The destination node replies with an ACK message for each transmission without error. All wireless sensor nodes are in each other's carrier sense range, thus eliminating the hidden node problem among nodes. Several collision avoidance techniques exist but they are not perfect to be integrated in the simulation. The channel capacity is 40 Kbps and data packets are 60 bytes in length. The wireless medium can accommodate up to 45 packets per second (pps), which is similar to what current wireless sensor nodes achieve. The other significant parameters of the simulations are shown in Table 6.1.

In the methods based on reinforcement learning, the initial learning and exploration

rates are $\alpha_0 = 0.1$ $\epsilon_0 = 0.2$, respectively. The rates are decayed according to the Darken-Chang-Moody algorithm [27] with parameter $\vartheta = 10^6$. The CMAC function approximator consists of ten tiles with two resolution elements. One CMAC of 20 resolution elements is needed for each action and each state component. The storage requirement in floating point numbers is 40 for RLCW and 80 for RLRATE, which are in the capacities of a sensor motes.

All the sensors enable active overhearing, which is a common assumption when the radio is never powered off. The broadcast nature of the communication enables the neighboring nodes to receive implicitly the notification and coordination messages. The data packets carry information such as buffer length q_p and the packets dropped D_p in RLCW and the incoming traffic λ in COCW.

6.1.2 Performance evaluation

Besides packet drop rate, aggregate throughput and total energy spent, composite measures are made: energy efficiency and fairness. Network fairness is applicable only for periodic workload.

For a network of n sensors, r_i packets belonging to the source i were collected at the sink. e_i Joules were consumed at i during the simulation run. The energy efficiency is given by the energy spent in the network for one packet received at the sink.

$$\textit{Energy efficiency} = \frac{\sum_{i=1}^{n} e_i}{\sum_{i=1}^{n} r_i} \tag{6.1}$$

The fairness metric employed is

$$\textit{Fairness} = \frac{\left(\sum_{i=1}^{n} r_i\right)^2}{n \sum_{i=1}^{n} r_i^2} \tag{6.2}$$

6.2 Non-periodic workload scenario

Event-driven sensing is the common example of non-periodic workload. Since the sensor nodes do not continually sense, fairness does not apply in this scenario. The totality of messages from the application is introduced into the network since rate regulation is not performed. Congestion is not completely removed but only alleviated. The buffers would overflow if the load of traffic exceeds the channel capacity.

To simulate the traffic generated by *random events*, a period of inactivity is introduced. An inactive source node does not generate packets, but can forward traffic. Transitions in activity are performed synchronously *every ten seconds* and a node has *one chance out of three* of being active in the next round.

The wireless sensor network topology is depicted in Figure 6.1. The network is evaluated under mean sensing rate ranging from 1 to 5 packets per second. A run lasts 2,000 simulated seconds. The results are then averaged over ten runs. We compare the methods in Table 6.2.

Table 6.2: Methods compared with non-periodic workload

RLCW	Contention regulation is performed by a policy obtained by reinforcement learning.
COCW	The second method is based on multi-agent coordination. The sensor nodes coordinate their contention levels.
BP+PM	Backpressure and Prioritized MAC techniques as in the work of Hull and Jamieson [4] is simulated. A congestion is notified whenever the buffer and channel occupancy exceeds the thresholds as in the reference, $q_{thres} = 0.75$ and $ch_{thres} = 0.85$. All sensor nodes use the same threshold values to detect congestion. Transmission is paused whenever a sensor node hears that its parent node is congested. Prioritized MAC grants a contention window divided by four to congested sensor nodes.
No control	The contention window is fixed as c_1.

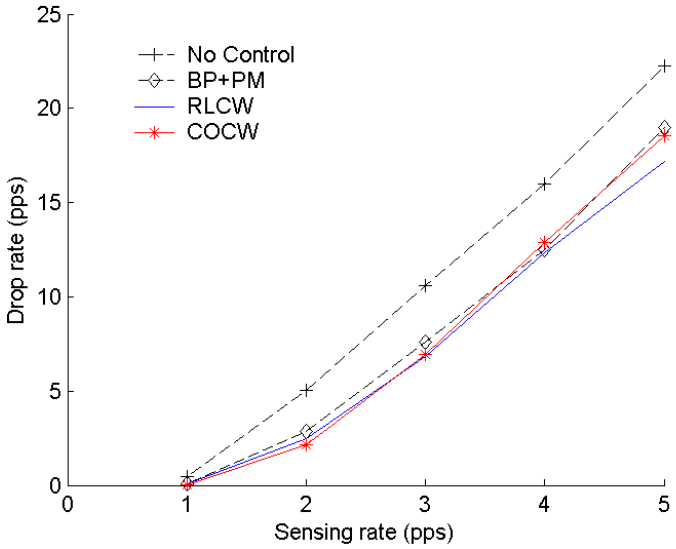

Figure 6.2: Total Packet drop in the network with non-periodic workload

6.2.1 Results of RLCW and COCW

Importance of gateway nodes:

Figure 6.2 shows the congestion of the network above 1pps as indicated by the dropping of packets. Congestion control is necessary because intermediate nodes at strategic locations affect the performance of the entire network. A gateway node near the sink consumes a lot of bandwidth since it forwards a lot of traffic. If a gateway node is congested, the aggregate throughput is diminished as showed at 2pps in Figure 6.3. Since the aggregate throughput is the most determining factor for energy efficiency, the ability of these gateway nodes to forward packet correctly is critical. Table 6.3 shows that node one and two are the gateways nodes in the topology.

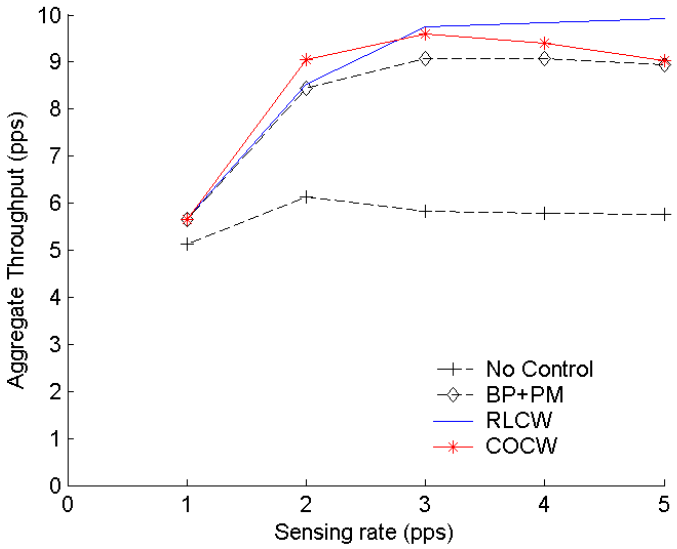

Figure 6.3: Aggregate Throughput with non-periodic workload

Table 6.3: Traffic forwarded by node

Subtree size	Node Id
9	1,2
4	6
2	11,13
1	3,7
0	4,5,8,9,10,14,15,16

Local channel contention:

It determines the transmission rate and thus the aggregate throughput. As shown in Figure 6.3, the learning method (RLCW) achieves up to 1 pps more throughput than the baseline method BP+PM.

Figure 6.4 shows that with the RLCW method, gateway nodes one and two always choose to use the short contention window. The remaining nodes use a short CW more frequently with COCW than with RLCW. Since there is less competition for channel with RLCW, gateways nodes can transmit more. Backpressure (BP) does not differentiate gateway nodes from leaf nodes since the congestion thresholds are identical for all nodes. BP implements the stop-and-go flow control (graph c). The gateway nodes choose the long CW about two times out of ten because their buffer levels can be low sometimes. The prioritized access (PM) decreases CW only for congested nodes. Since gateway nodes forward large amount traffic, a weak channel contention leads to the reduction of throughput as observed with the BP+PM method.

Energy efficiency:

RLCW consumes less energy in the network than COCW and BP+PM as result of moderate contention. In addition, more packets are received at the sink with the RLCW and COCW methods, reducing the energy spent per packet. As shown in Figure 6.5, RLCW is the most energy efficient, outperforming BP+PM by 14% (-0.007J/packet).

The throughput and efficiency of COCW declines with the report rate because of communication delays. Since the parent node becomes more congested with increasing sensing rate, some child nodes do not receive their coordination message. It is interesting to note that less energy is spent when no method is used (Figure 6.6). The total energy spent is reduced because of higher drop rates and fewer packets to forward in the end.

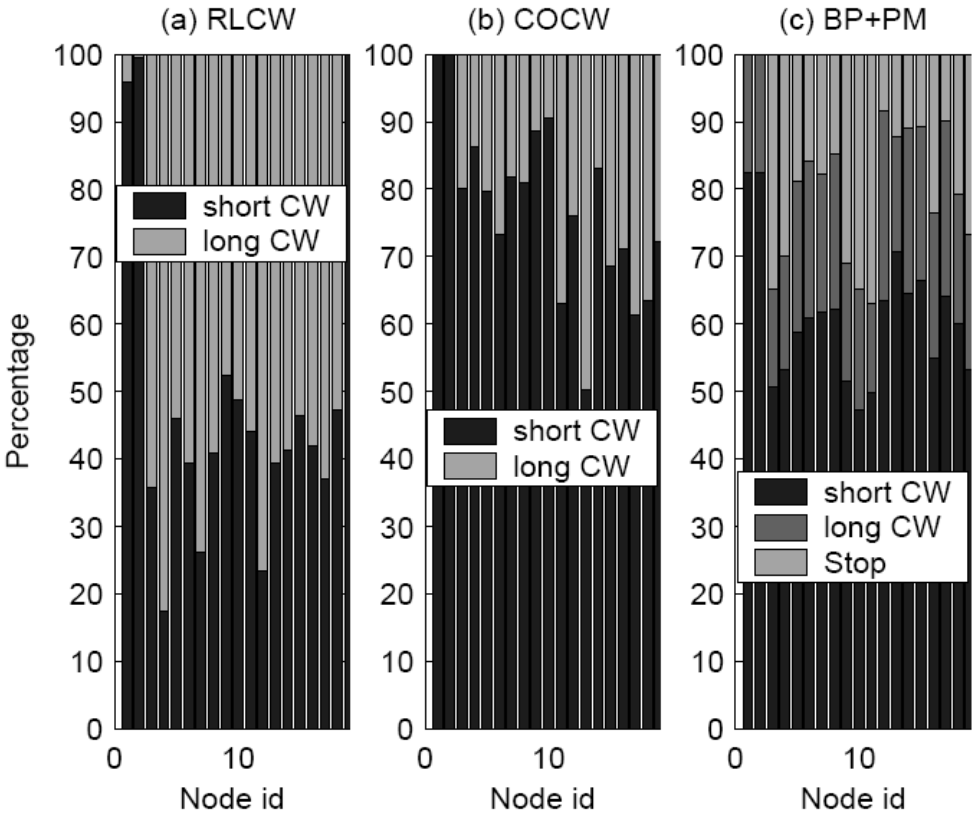

Figure 6.4: Contention window chosen by sensor nodes at 1pps

6.2.2 Analysis of the value functions and learned policy

To prove the convergence of the SMART algorithm in the contention window regulation problem, the value functions and the policy are analyzed.

Figure 6.7 shows the value function and the maximizing action for a given state and at a given node in the case of RLCW. A rise in the cost is observed within the first 500 seconds due to the absence of policy. In addition, the agent explores actions and chooses arbitrarily between action 0 and 1. After 500s, only one action is selected, validating policy convergence. The following exploitation of the policy stabilizes progressively the value function. The value converges to its minimal value as it is a cost.

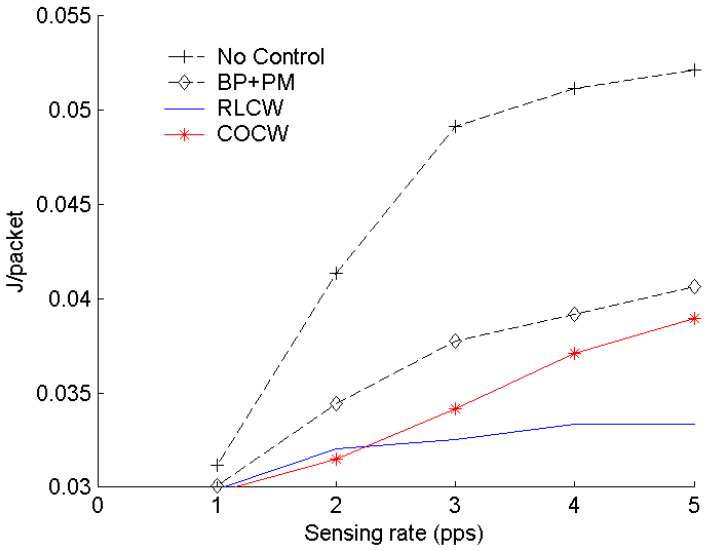

Figure 6.5: Energy efficiency with non-periodic workload

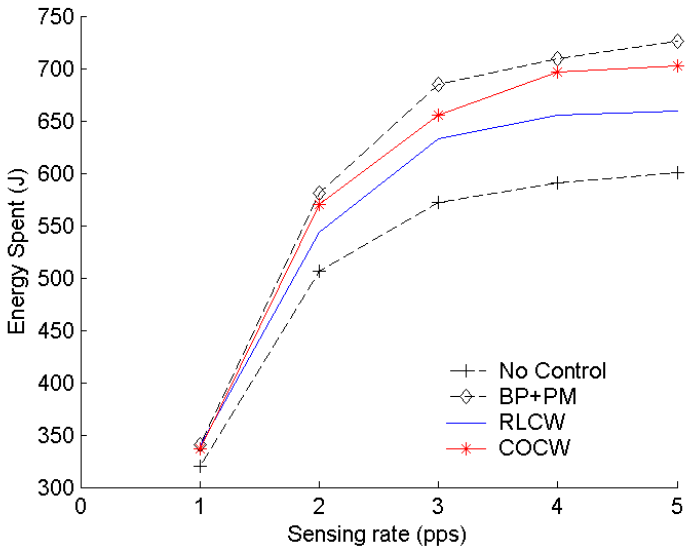

Figure 6.6: Total energy spent in the network with non-periodic workload

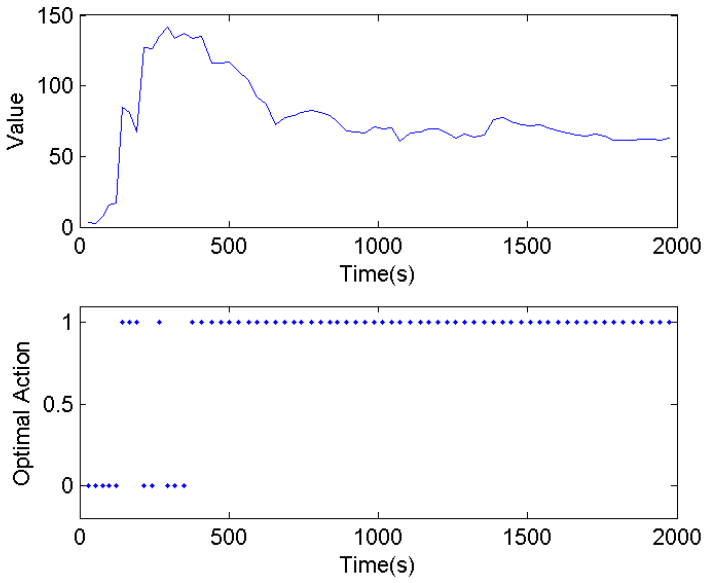

Figure 6.7: Convergence of the value function and the policy

Figure 6.8 plots the value and the policy at node seven and 15 which differs in the degree of their parent node. These value functions were collected after 2,000 seconds of learning at sensing rate 1pps. Lower values are desirable because they represent costs.

The value function displayed in graph (a) indicates the existence of notification delay. If the notification was instantaneous, the cost would concentrate at occupancy ($q_p = 1.0$). However the cost at intermediate occupancy levels ($q_p \approx 0.5$) suggests that buffer overflows occurred although the buffer was observed 50% full. Because of the notification delay, the state is not fully observable. The uncertainty in the observation is leveraged by learning the expected outcomes.

As illustrated in graphs (a) and (c), node seven receives more cost than node 15 as its parent node is a gateway node which is prone to congestion. These observations in addition to the favorable results of RLCW support the need for adaptive congestion avoidance in wireless sensor networks. In the policy graph (b), the node seven learns

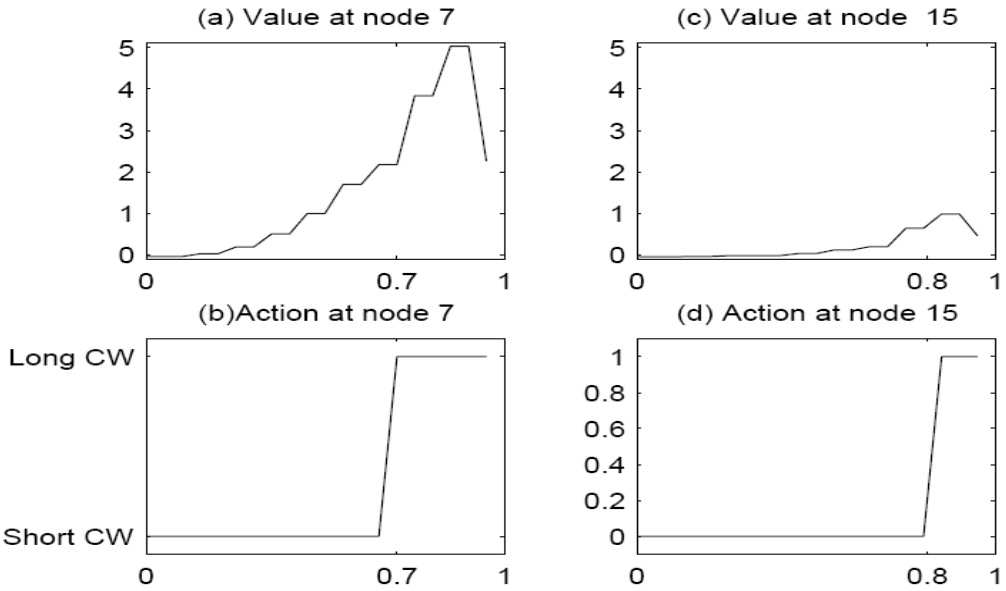

Figure 6.8: Value function and Action learnt in RLCW as function of buffer occupancy q_p

to use a long contention window when the occupancy is above 70%. The node 15 has learned that its parent node is less prone to congestion and thus set the threshold to 80% (cf. graph (d)). This observation verifies that nodes can individually learn the policy adapted to the environment.

6.3 Periodic workload scenario

In this experiment, all sensor nodes generate packets periodically without interruption. The bandwidth can be allocated fairly since the number of source nodes is constant. Rate regulation is performed at the messaging layer before the packets are admitted in the network. The control agent may reject the admission of new packets in order to reduce the packet generation rate (i.e. $<$ sensing rate).

The congestion control schemes are compared under sensing rate varied from 0 to

2.5 packets per second on the same topology. All the sensors are generating packets. Each experiment is repeated ten times and then the results are averaged. Each run lasts 2,000 simulated seconds. The contention window is not controlled and left to 31ms in the compared methods. Table 6.4 summarizes the rate regulation methods.

6.3.1 Results of RLRATE and CORATE

Effect of a Lossy channel

The rate control reduces the packets admitted into the network to mitigate congestion. As shown in Figure 6.9, reducing the rate directly at the source removes almost entirely the packet drops.

In this scenario, RLRATE drops less packets than Fusion. The source rate limiting (RATE) in Fusion relies on counting all packets transmitted by the parent node including the corrupted ones. With a lossy channel, the retransmissions skew the traffic shaping performed by the token bucket. As a result, Fusion generates packets in excess which later congest the traffic. In contrast, the two proposed methods detect unfairness by

Table 6.4: Methods compared with a periodic workload

RLRATE	The agent learns the rate control policy to minimize unfairness.
CORATE	The agents coordinate their rate decisions to minimize unfairness.
Fusion	RATE+BP+PM: The combination of three techniques, Fusion [4], comprises source rate limiting, hop-by-hop backpressure and prioritized MAC. The traffic of the parent node is measured and the local rate is adjusted with a token bucket. This simple passive method limits the sensor to send at the same rate of its descendent. The token generation rate is equal to μ_p/N_p where μ_p is the transmission rate of the parent node and N_p its subtree size.
No control	All sensing packets are admitted into the network.

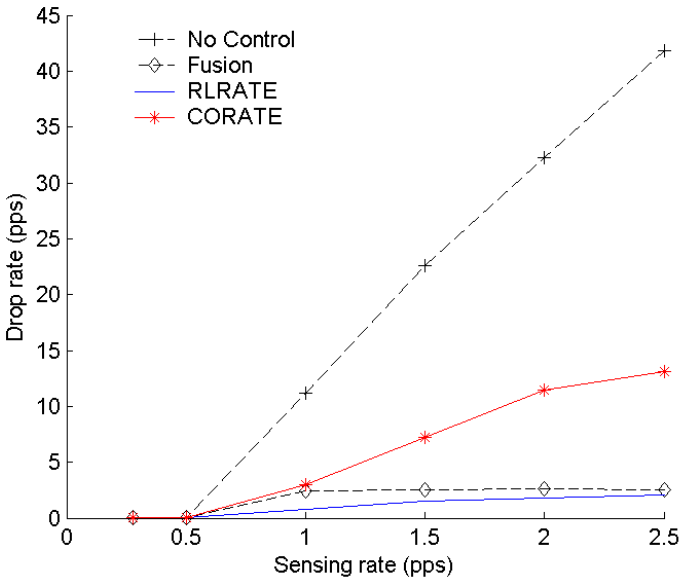

Figure 6.9: Total Packet drops in the network with periodic workload

counting the packets generated and transmitted by the parent node. The channel quality has less negative effects on the performance since only the successful transmissions are counted.

Effect of increased decision rate:

CORATE suffers from the notification delay which reduces responsiveness of the agent. As seen in Figure 6.9, the drop rate in CORATE increases with the offered load. Coordination with the Min-Sum algorithm is based on message passings. Due to the contention delay, the parent node sends one coordination message only after half of the neighboring sensor nodes have transmitted in average. Because some important messages are received after the actual decision making, packets are admitted in excess. Each packet generation is a decision with CORATE, therefore buffer overflows increases with the sensing rate as shown in Figure 6.9. Figure 6.12 also indicates reduced aggregate throughput for CORATE as consequence of congestion. The frequency of messages updates is limited

by the channel capacity. Therefore, more erroneous decisions are made when the sensing rate increases.

As illustrated in Figure 6.10, RLRATE behaves as expected when the sensing rate is increased from 1pps to 2pps. On the contrary, CORATE does not halve the packets generated rate like RLRATE, and causes overload traffic.

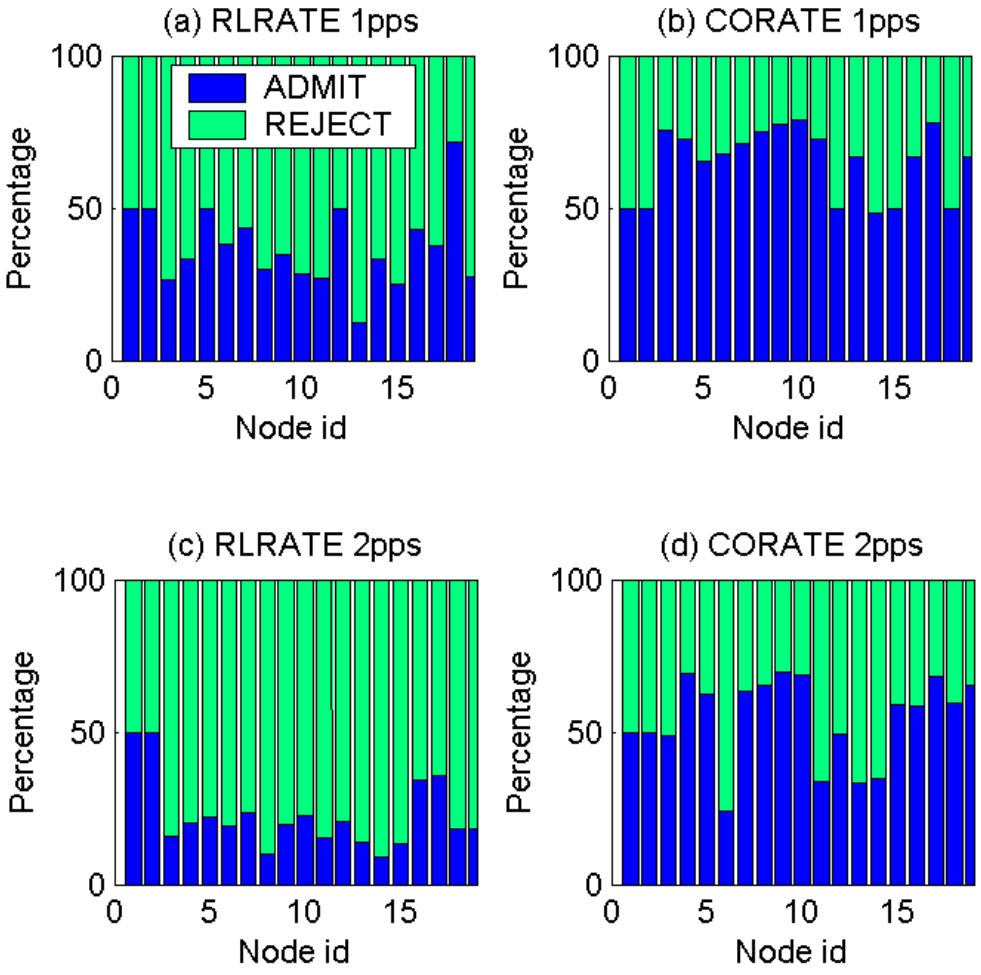

Figure 6.10: Decisions taken by greedy node

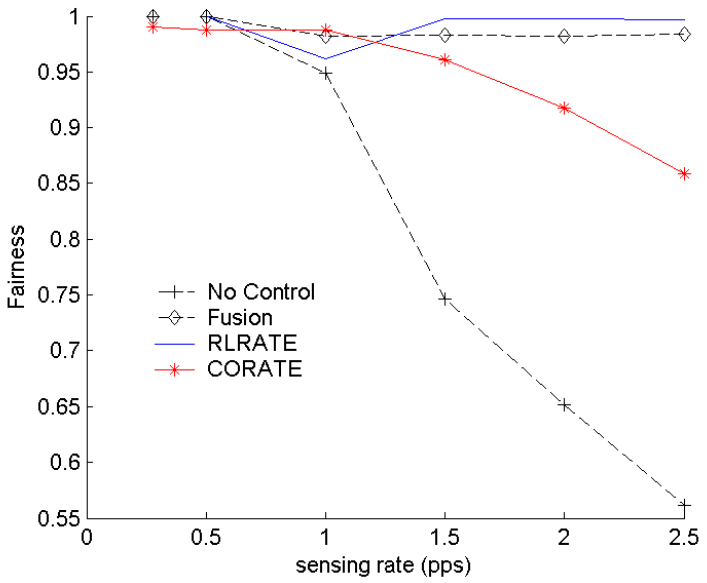

Figure 6.11: Network Fairness with periodic workload

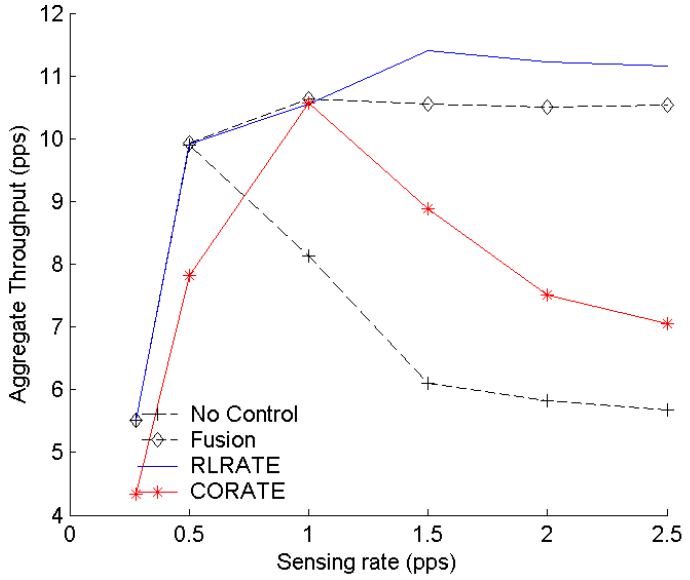

Figure 6.12: Aggregate Throughput with periodic workload

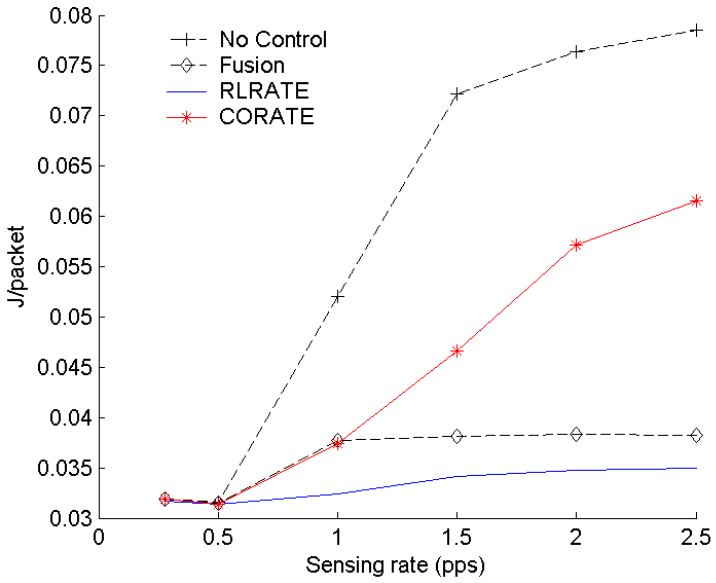

Figure 6.13: Energy efficiency with periodic workload

Fairness

Figure 6.11 compares the fairness provided by the rate control methods. RLRATE and FUSION allocate fairly the bandwidth and the network is able to keep fairness above 0.95 for all values of sensing rates. The discrepancy of the results of CORATE is also shown at the fairness which declines with the sensing rate.

RLRATE yields fairness similar to Fusion. However, RLRATE incurs the least packet drops. RLRATE provides fairness because the reward was allocated to penalize greedy actions. The convergence of the value and policy are similar to that described in the non-periodic workload.

Energy Efficiency

Figure 6.13 reports the energy consumed in the network for one packet received at the sink. Above 1.5pps, RLRATE outperforms in throughput all the other methods. RLRATE is the most energy efficient scheme by reducing consumption up to 0.005J/packet

(-15%) compared to Fusion.

The CORATE method exhibits as expected a decreasing efficiency with the increasing sensing rate. The best results are obtained with RLRATE in particular because the state and reward are available locally without resorting to communication.

6.4 Discussion

The aim of this thesis is to investigate the possible benefits of coordination and learning in congestion control. Two issues were raised: notification delay and unpredictable traffic pattern.

The use of reinforcement learning is a well-adapted option to adjust automatically the detection and control parameters of wireless sensor nodes. The uncertainty in the states is alleviated with the prediction features of the Semi-MDP model. The notification delay and the unpredictability of congestion can be circumvented with value functions which represent the long term expected utility. The analysis of the policy showed that agents learn different policies in order to adapt to each situation. The baseline method using threshold incurred more drops than with the learning method. Coordination is achieved by using reward or state related to multiple agents. One drawback of RL methods can be the initial learning phase, but which took only 10 minutes in simulation time.

With the coordination algorithm Min-Sum, the global utility is decomposed into several local utilities related to smaller parts of the system. The coordination graph is depicted with spanning tree. The interactions are in practice more complex and more difficult to determine in a wireless context. One difficulty is to establish formalized relationships between agents, which are sometimes not straightforward.

Congestion avoidance is modelled as the coordination between several agents in COCW. Coordination functions are loose upper bounds of packet loss. They are based

on heuristics rather than rigorous analysis. COCW outperformed Backpressure. This illustrates the strength of the Min-Sum algorithm. In the experiments, the data traffic provides *free transport* for messages. The message passing has to accelerate if decision intervals shorten. In reason of communication delay, real-time decision making is not achievable. Results showed that CORATE suffers from this delay and its performance deteriorates with increasing sensing rates.

Simulations results showed that reinforcement learning based methods enhanced the network efficiency and fairness. The coordination methods were limited by the communication delay and may not be good solutions for fast paced decision problems. A summary in given in Table 6.5

Regarding the computational efforts, the coordination algorithm requires a dozen of operation including memory access for each packet sent and each packet overheard (from neighbors). The learning algorithm requires a dozen of elementary float operation for each packet to send. For the coordination a method, the drop minimization is not performed well when the channel is congested. The delay to receive control information impedes the correctness of decision making. A congestion affects coordination algorithm and which in turn loses efficiency.

Learning methods are scalable since the learning process only involves the local sensor, the parent sensor node and the immediate child sensor nodes. Therefore, learning methods scale easily. The Chapter 4, section 3.3 mentions the difficulties encountered to make the learning algorithm converge.

Coordination methods also scale up easily since the messages are inserted in the header and tolerant to asynchronous transmission. The problem of minimizing congestion is transformed into smaller coordination problem. The coordination is only local,

because the causes of congestion and consequences are local. Chapter 5, section 3.4 mentions implementations issues and overhead induced by the scheme.

The processing is at sending time. There are a dozen of arithmetic operations in floating numbers (cf. Appendix A.1). The methods may apply to very large sensor networks since the communication overhead does not grow with the network size. In comparison with FUSION, communication overheads are strictly similar because the proposed methods piggyback control data as well. Fusion also uses active listening of all packets. The proposed methods use in average less than 100 bytes for storage and less than ten arithmetic operations for each packet sent.

This study was performed on a simulator rather than on real sensor motes. Hidden node problems were neglected with the simulated network because this work first addresses congestion regardless of the underlaying MAC. The RTS/CTS control packets or delays should be used if collisions prevent communications. A simple packet error rate was simulated to determine the robustness in realistic conditions. The set of contention windows was limited to two, because a larger set would increase the convergence time and the size of messages without significant benefit.

A real life user trial would require the following settings:

- for each packet transmit, a parameter modulates the contention or the probability of real transmission. In 802.11 DCF, the contention window has this purpose. It can also be the slot allocation or transmissions schedule allocation.

- a MAC layer which allows to extract information from the header of packets which are addressed to its receiver. Since the control information is communicated within the data packets, the neighbor nodes must be in listening mode when the relevant control information is sent. In 802.11 based Mac, the receiver is always listening so the proposed methods work fine. In S-MAC, parent and child nodes stay connected once they exchanged their schedules. Thus, the learning and coordination process

are not affected by choosing the one of the awaken neighbors as destination of control information.

- a routing algorithm which does not modify frequently the topology and connectivity. The forwarding sensor has to be invariant for every sensor because the learning process is specific to this forwarding node. The communication structure has to stay stable even in presence of sleep schedule (or radio turned off).

Table 6.5: Summary of the studied methods

Method	Strengths	Weaknesses
Reinforcement learning	No need to determine the coordination model. Can adapt to imperfection such as communication delay	The initial learning phase prevents immediate exploitation of the policy
RLCW	Best efficiency. Gateway nodes have priority.	State is not available directly
RLRATE	Most Fair. Highest throughput	Learn an intuitive policy
Multi-agent coordination	Take directly the optimal action by estimating decision with real values.	Expensive in communication. The coordination model is to be determined (or heuristic).
COCW	Good Efficiency	Efficiency declines with increasing rates
CORATE	Fair	Efficiency and Fairness decline severely with increasing rates

Chapter 7

Conclusions

7.1 Contributions

This thesis addresses the congestion issue with artificial intelligence methods. Conventional methods do not address the issue of notification delay in contention based MAC. Congestion likelihood is variable and its notification experiences delay. Such delay is the root of the slow responsiveness and inaccuracy of most existing congestion control schemes.

Instead of controlling congestion using a fixed algorithm, this thesis proposes to use the learning approach to alleviate congestion in sensor networks. The congestion having uncertain outcomes, the learning method provides ways of enhancing the throughput. Packet drop and unfairness have been addressed.

Results confirm that buffer levels have different interpretation in terms of drop probability. The algorithm works because there is a direct relationship between buffer state and buffer overflow. The control policy is learned iteratively and outperforms existing methods. The proposed learning approach promises more flexibility and better performance.

An alternative approach based on the 'Min-Sum' algorithm is used to coordinate

packet transmission and then solve the congestion control problem. One conclusion is that the coordination algorithm produces bad results as the network load increases. The reason is that at congestion time, packets containing control information are lost or delayed.

In all scenarios, the methods based on learning are 15% more energy efficient than an existing scheme Fusion. In real application, this improvement means that the lifetime of the network is improved for same traffic pattern, same activity and same packets collected at the base station. A network running the proposed method delivers about twice the throughput and expends less energy compared to a network without any congestion control.

The superiority of the proposed solutions over method such as FUSION is most discernible when the sensing rate is high (higher than 1 pps). For most applications of the wireless sensor networks, the sensing rate is pretty low. Some applications of such high sensing rate include camera sensors or motion detectors.

In low duty cycle sensor network, congestion can appears because constrained connectivity or sleep schedule. The proposed schemes work without any modification since they do not depend on a particular medium access protocol. S-mac or B-mac is a good candidate for such applications.

7.2 Applications and Implementation

In all collection scenarios, the methods based on learning are 15% more energy efficient than an existing scheme FUSION. In real application, this improvement means that the lifetime of the network is improved for same traffic pattern, same activity and same packets collected at the base station. A network running the proposed method delivers about twice the throughput and expends less energy compared to a network without

any congestion control.

The superiority of the proposed solutions over method such as FUSION is most pronounced when the sensing rate is high (higher than 1 pps). For most applications of the wireless sensor networks, the sensing rate is pretty low. Some applications of such high sensing rate include camera sensors or motion detectors. In low duty cycle sensor network, congestion can appears because constrained connectivity or sleep schedule. The proposed schemes work without any modification since they do not depend on a particular medium access protocol. S-mac or B-mac is a good candidate for such applications.

Learning methods are scalable since the learning process only involves the local sensor, the parent sensor node and the immediate child sensor nodes. The proposed methods use data packet to piggy back the control information. Therefore there is no communication overhead. A real life user trial would require that a MAC layer allows packet header snooping and that the contention can be tuned for each packet. Also, the paths towards the sink has to be invariant from one point. Those requirements points are discussed in the discussions section of the Chapter 6.

In the learning scheme RLCW, the state is defined as the queue length of the parent node. The attempt to add the channel load and local queue as state component was unsuccessful. The extra state components presented large deviation due to their sampling. Therefore the generalization done by the CMAC had undesirable affect on the policy.

Initially, it was difficult to find arguments supporting the statement that the underlying process is a Markov chain. The theoretical foundations then were not very strong unless a continuous time model which is more flexible. Semi Markov decision processes allow us to apply reinforcement learning on the congestion control problem. The decision times are random and multiple state transitions can occur between two steps. The reader is invited to refer to Chapter 4, for more details in implementation complexities.

7.3 Future work

The thesis is implicitly concerned with data collection in tree communication structure. In the periodic scenario, it was implied that all the sensors are sensing and transmitting. However, some applications require just a sparse population of sensors to perform the sensing task. The RLRATE, CORATE methods will not alleviate unfairness as the downstream fairness index requires the parent node to generates new packets. If the parent is not a data source, then the index will fail in equalizing the generation rates of upstream sensors. Another way to improve multi-agent coordination is by extending it over communication range so that distant sources are allocated a fair share of bandwidth.

In the simplified view of analysis, the results are intuitive and still give insight to the problem. There is a future work is formalizing the congestion control problem more rigorously, as a SMDP. A queuing model could be envisaged once the transition probability is parameterized. If more variables are introduced; the model would validate make the simulation results easier to interpret. In the current state of work, learning from experience produces a decision maker with relatively good accuracy. A future model has to trade off between strength of foundations and ease of implementation. Although the proposed methods are still independent from the MAC layer, a thorough analysis of the congestion can be done only with a particular MAC. This thesis studies adaptive approaches which are generic enough to apply to other wireless sensor communication systems.

Appendix A

Algorithms

A.1 SMART Algorithm

Initialize initial time $t = 0$, last event time $t_l = 0$, total time $T = 0$ and the reward rate $\rho = 0$, state s and s', action values $R_{old}(s, a) = R_{new}(s, a) = 0$ for all $s \in S$ and $a \in A(s)$

Set learning rate α, averaging rate β, exploration rate ϵ

for *(event e)* **do**

  Determine the new state s'

  Determine the cost and the transition time $\tau(s, a)$

  Calculate the reward $r(s, a)$ and the average reward ρ

  Calculate the TD error

$$\Delta R = r(s, a) - \rho \cdot \tau(s, a) + \max_{b \in A} R_{old}(s', b) - R_{old}(s, a)$$

  Update the State-Action Values

$$R_{new}(s, a) = R_{old}(s, a) + \alpha_t \cdot \Delta R$$

  Select the action a which is the greedy action with probability $1 - \epsilon$, otherwise select a random action

$$R_{old}(s, a) \leftarrow R_{new}(s, a) \quad \forall (s, a) \in S \times A$$

  Update the reward rate $\rho \leftarrow \frac{\rho * T + r(s,a)}{T + \tau(s,a)}$

  Update the last event time $t_l \leftarrow t$, the current state $s \leftarrow s'$ and the total time $T \leftarrow T + \tau(s, a)$

  Update learning rate α, averaging rate β, exploration rate ϵ

  Perform the action a

end

Algorithm 5: Learning a policy with the SMART Algorithm

A.2 Min-Sum Algorithm

Discover neighbors nodes of node i in the spanning tree: $N(i)$={parent+child nodes}

Construct and update the potential functions $\psi_i(s_i, a_i)$ and $\psi_{ij}(s_i, s_j, a_i, a_j)$ for every neighbor $j \in N(i)$

for *each packet to transmit* **do**

  Select a neighbor j in Round Robin

  Calculate the message to node j

$$m_{ij}(a_j) = \min_{a_i}\{\psi_i(s_i, a_i) + \psi_{ij}(s_i, s_j, a_i, a_j) + \sum_{k \in N(i)\setminus j} m_{kj}(a_i)\}$$

  Calculate

$$g_i(a_i) = \psi_i(a_i) + \sum_{k \in N(i)} m_{ki}(a_i)$$

  Choose action $a_i^* \in a_0, \ldots, a_M$

$$a_i^* = \arg\min_{a_i} g_i(a_i)$$

  Transmit the packet with information

$$[i, s_i, j, m_{ij}(a_0), \ldots, m_{ij}(a_M)]$$

end

for *all packets overheard with* $j == i$ **do**

  save and update [j, s_j, i, $m_{ji}(a_0)$, ...] in Storage

end

Algorithm 6: Distributed Coordination with Min-Sum

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Amsterdam, Netherlands: 1999)*, 38(4):393–422, 2002.

[2] Alec Woo and David E. Culler. A transmission control scheme for media access in sensor networks. In *Mobile Computing and Networking*, pages 221–235, 2001.

[3] C. Y. Wan, S. B. Eisenman, and Andrew T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *SenSys*, pages 266–279, 2003.

[4] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *ACM SenSys 2004*, Baltimore, MD, November 2004.

[5] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. Esrt: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc*, pages 177–188, 2003.

[6] Cheng Tien Ee and Ruzena Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *SenSys*, pages 148–161, 2004.

[7] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis. Interference-aware fair rate control in wireless sensor networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):63–74, 2006.

[8] Chonggang Wang, Kazem Sohraby, Victor Lawrence, Bo Li, and Yueming Hu. Priority-based congestion control in wireless sensor networks. In *SUTC '06: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06)*, pages 22–31, Washington, DC, USA, 2006. IEEE Computer Society.

[9] Na Yang. Congestion avoidance based on lightweight buffer management in sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(9):934–946, 2006. Member-Shigang Chen.

[10] M. Zawodniok and S. Jagannathan. Predictive congestion control mac protocol for wireless sensor networks. In *Internation Conference on Control and Autonation, ICCA '05*, pages 185–190, 2005.

[11] Chieh-Yih Wan, Shane B. Eisenman, Andrew T. Campbell, and Jon Crowcroft. Siphon: overload traffic management using multi-radio virtual sinks in sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 116–129, New York, NY, USA, 2005. ACM Press.

[12] Douglas H. Norrie Weiming Shen and Jean-Paul Barths. *Multi-agent systems for concurrent intelligent design and manufacturing.* Taylor & Francis Group, 2001.

[13] R. S. Sutton and A. G. Barto. *An introduction to reinforcement learning.* Cambridge, MA, USA: MIT Press, 1998.

[14] T. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck. Solving semi-markov decision problems using average reward reinforcement learning, 1999.

[15] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Auton. Robots*, 8(3):345–383, 2000.

[16] Jeff G. Schneider, Weng-Keen Wong, Andrew W. Moore, and Martin A. Riedmiller. Distributed value functions. In *ICML*, pages 371–378, 1999.

[17] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *J. Artif. Intell. Res. (JAIR)*, 19:399–468, 2003.

[18] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, pages 227–234, 2002.

[19] Kao-Shing Hwang, Cheng-Shong Wu, and Hui-Kai Su. Reinforcement learning cooperative congestion control for multimedia networks. In *International Conference on Information Acquisition*, 2005.

[20] C. K. Tham. Online function approximation for scaling up reinforcement learning, 1994.

[21] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, 1988.

[22] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[23] Jelle R. Kok and Nikos A. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *BNAIC*, pages 359–360, 2005.

[24] Christopher Crick and Avi Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 159–16, San Francisco, CA, 2003. Morgan Kaufmann.

[25] Mark A. Paskin, Carlos Guestrin, and Jim McFadden. A robust architecture for distributed inference in sensor networks. In *IPSN*, pages 55–62, 2005.

[26] The network simulator ns-2 http://www.isis.edu/nsnam/ns/.

[27] Christian Darken, Joseph Chang, and John Moody. Learning rate schedules for faster stochastic gradient search. In *Proc. Neural Networks for Signal Processing 2*. IEEE Press, 1992.