Investigating Performance Improvements in Wireless Networks Using Probabilistic Graphical Models

by

William Sivert Rörich Pretorius

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering (Electrical & Electronic) in the Faculty of Engineering at Stellenbosch University

Supervisor:	Prof. J.A. du Preez
Co-supervisor:	Dr. R. Wolhuter

December 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....December 2016. .

Abstract

Investigating Performance Improvements in Wireless Networks Using Probabilistic Graphical Models

W.S.R. Pretorius

Department of Electrical & Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MScEng (E&E)

December 2016

Probabilistic Graphical Models (PGMs) have proven to be a powerful and effective tool for predicting the behaviour of probabilistic systems. Their applicability for improving the performance of wireless networks, where most strategies are probabilistically founded, is therefore worth exploring. PGMs can infer states and conditions within the network and allow protocols to act accordingly. However, as this implies decision-making under uncertainty, investigating the application of PGMs for this purpose would have merit.

In this work, we create an effective method for making decisions under uncertainty by expanding the current theory of strong junction trees to allow for loopy decision cluster graphs. However, similarly to the behaviour of loopy cluster graphs, this method also leads to imprecise probabilities and utilities, and sub-optimal decision strategies.

We created 3 PGM-augmented Round Robin Medium Access Control (MAC) protocols by using different PGMs to determine which slave node the master node should poll next. This resulted in reduced latency for packets during unequal traffic loads. Furthermore, we created a PGM-augmented Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC protocol by using a PGM in order to estimate the number of contending nodes and allowing the node to change the length of its contention window accordingly. This resulted in an efficient and fair network protocol irrespective of the number of nodes in the network.

Uittreksel

Ondersoek na Verbeteringe in Radionetwerke met behulp van Probabilistiese Grafiese Modelle

("Investigating Performance Improvements in Wireless Networks Using Probabilistic Graphical Models")

W.S.R. Pretorius

Departement Elektries & Elektronies Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MScIng (E&E)

Desember 2016

Probabilistiese Grafiese Modelle (PGM'e) het reeds bewys dat dit 'n kragtige en doeltreffende manier bied om die gedrag van probabilistiese stelsels te voorspel. Dit blyk dus aantreklik om hul toepassing in radionetwerke, waar strategië probabilisites van aard is, te ondersoek. PGM'e kan die toestande en omstandighede van die netwerk afskat en protokolle kan daarvolgens optree. Aangesien dit egter impliseer dat daar besluite tydens onsekerheid geneem moet word, is die ondersoek om PGM'e te gebruik vir hierdie doel ook van belang.

In hierdie werk skep ons 'n effektiewe metode om besluite tydens onsekerheid te neem deur die huidige teorie van sterk aansluitingsbome uit te brei om beslissing-kluster-grafieke met lusse moontlik te maak. Soortgelyk aan kluster-grafieke, lewer hierdie metode egter onakkurate waarskynlikhede, nut-waardes en sub-optimale beslissing-strategië.

Ons het 3 PGM-uitgebreide 'Round Robin' medium toegangsbeheer-protokolle geskep deur verskillende PGM'e te gebruik om te bepaal watter slaaf-node die meester-node volgende moet ondervra. Hierdie lei tot verkorte transmissievertragings van pakkies tydens ongelyke netwerkladings. Verder het ons 'n PGM-uitgebreide 'CSMA/CA' medium toegangsbeheer-protokol geskep deur 'n PGM te gebruik om af te skat hoeveel nodusse aan die kontensie wil deelneem en die lengte van die kontensie-venster daarvolgens aan te pas. Hierdie lei tot 'n meer doeltreffende en billike netwerk, ongeag die aantal nodusse in die netwerk.

Acknowledgements

I would like to thank our Heavenly Father who gave me the insight and strength to finish this project.

I would also like to express my sincere gratitude to the following people:

- Prof. J.A. du Preez and Dr. R. Wolhuter for their excellent guidance and support.
- My parents, for their love and support.
- All my friends and colleagues for their help and support.

Contents

List of Figures

List of Tables

Nomenclature

Network Acronyms

CSMA/CA	Carrier sense multiple access, collision avoidance
CW	Contention window size
MAC	Medium access control
N	Number of nodes

Statistical Acronyms

CG	Cluster graph
JT	Junction tree
LIMID	Limited memory influence diagram
PGM	Probabilistic graphical model
RV	Random variable
γ	Decision potential
μ	Utility factor
ρ	Probability factor

Random Variables

CP_i	Is node i participating in the contention period
CQ_i	Current queue length of node i
DR_i	Data rate of node i
EX_i	Expecting activity from node i
MRD_i	Match the number of RTS and data messages of node i
NC_i	Overall condition of Node i
NP_i	Is node i being polled
NQ_i	New queue length of node i
NS_i	Is node i successful at winning access to the channel
$NTSP_i$	New time elapsed since node i was polled

PID	Identity of the polled node
RGR_i	Rate at which RTS-messages (of node i) are generated
SC	Is contention happening successfully in the network
dif_i	Difference between number of RTS and data messages sent by node i
nrs_i	Number of RTS messages sent by node i
pqs_i	Previous queue length sent by node i
pwt_i	Previous waiting time of node i
tsp_i	Time elapsed since node i was polled
tst_i	Time elapsed since previous transmission of node i

Chapter 1

Introduction

1.1 Motivation

Wireless networks are extremely common and have, therefore, received much attention in order to improve their performance. Such research has focused on a number of different aspects, such as the physical-, Medium-Access-Control(MAC)- and network layers. As far as the latter two are concerned, most of the strategies are probabilistically founded and it is logical that this should be kept in mind in any effort aimed at performance improvement in those layers. Probabilistic Graphical Models (PGMs) have proven to be a powerful and effective tool for predicting the behaviour of probabilistic systems and assisting with their subsequent optimisation. It thus appears attractive to investigate their applicability to wireless networks. PGMs can infer states and conditions within the network and allow wireless protocols to act accordingly, creating a more adaptive and efficient strategy. As responding to system information implies decision-making -- and, more specifically in this case -- decision-making under uncertainty, it was felt that an investigation of the application of PGMs for this purpose would have merit.

1.2 Background

1.2.1 Probabilistic Graphical Models: Cluster Graphs

There are uncertainties within wireless networks -- for example, which nodes want to transmit data. Estimating the probabilities of these uncertainties can allow network protocols to adapt accordingly, creating an efficient strategy. Therefore, an efficient method for calculating these probabilities is necessary. Cluster graphs (a PGM) are an ideal tool for this, since they are capable of calculating the probabilities of random variables (RVs) efficiently by exploiting Bayes' rule [1, 2].

Cluster graphs work as follows:

- Probability information about RVs is stored in tables (called probability factors). These probability factors form cluster nodes (labelled with the corresponding RVs) in the graph.
- These cluster nodes are connected to each other with separation sets (*sep-sets*), the overlapping RVs of the two cluster nodes. Cluster nodes that have no mutual RVs are not connected.
- Since observed RVs are variables whose states are known, the effects they have on the remaining variables are calculated and stored, at which point the observed RVs are removed from the cluster graph.
- Messages (a probability factor containing the RVs of the sep-set) are sent between the cluster nodes. These messages are propagated until their values converge. Note that convergence is not guaranteed.
- Sending the messages requires that the following operations on probability factors be defined: absorb, cancel, marginalize, and normalize.
- Once the messages converge, the probabilities of RVs are calculated using a single cluster node and the adjacent sep-sets containing the queried variable(s). Note that the probabilities are not guaranteed to be exact.

If the cluster nodes are connected in such a way that no loops are formed, the cluster graph is also known as a junction tree (JT). A JT can be created by a process of eliminating variables [1, 2]. The order in which variables are eliminated determine the structure of the graph, as well as the number of RVs in each cluster node. Thus a loopy cluster graph can be represented with more than one JT. Since the number of RVs in a cluster node determine the size of the table, an optimal order in which to eliminate RVs exist. However, determining the optimal order is NP-hard.

Junction trees result in exact calculations and always converge with one set of message passing. Since messages are only sent once, the normalization operation is not strictly necessary. However, JTs typically have more RVs in each cluster node, which results in an exponential growth in the size of the table with each extra RV. This exponential growth makes JTs impractical for problems with a significant number of RVs.

Even though loopy cluster graphs are imprecise (converging to incorrect probabilities), the results they give are typically close to the correct solution. Also, the significantly reduced size of the probability factors makes them a great deal more practical.

1.2.2 Decision Theory

Responding to system information implies decision-making and since there is uncertainty in wireless networks, decision-making under uncertainty is necessary. The decision theory framework provides a foundation for the decision-making task. Decisions are represented as decision variables to which the user can assign probabilities (also know as a decision strategy). A decision can rely on both RVs and other decision variables, and in these cases the decision strategy would specify which action to take given the input variables [1, 2, 3].

In order to make decisions, the system requires information about how preferable the outcomes of certain variables are. This is achieved by assigning numerical values (also known as utilities) to the outcomes of certain variables (this information can be stored in a table known as a utility factor). These assignments allow us to define an expected utility, stating how preferable a situation is when there are uncertainties regarding the relevant RVs. Decisions are made in order to maximize the expected utility, creating the most preferred situation [2, 3, 4].

1.2.3 Medium Access Control in Wireless Networks

We focused on improving the Medium Access Control layer of wireless networks; this is the layer responsible for allowing nodes to access the medium. The protocols that were investigated are the Round Robin and Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC protocols.

1.2.3.1 Round Robin MAC protocol

The Round Robin MAC protocol appoints a network node as a master node. This node polls the other nodes (called slave nodes) in the network sequentially, granting them permission to transmit data [5].

Since the master node controls the medium, no collisions in the medium (2 or more nodes transmitting at the same time whose transmissions interfere with each other) can occur, making this protocol efficient under high traffic loads. However, the protocol is inefficient at low traffic loads; a node with data to transmit has to wait to be polled, increasing the transmission delay (latency) of data packets.

We investigated this protocol due to its simplicity and its potential improvement by having an adaptive polling order.

1.2.3.2 CSMA/CA MAC protocol

The CSMA/CA MAC protocol allows network nodes to contend for access to the medium. For a node to transmit data over the channel, it has to win channel access via contention. Contention is resolved by allowing participating nodes to randomly choose an idle time slot (between 0 and CW, the contention window) by transmitting a Request-To-Send (RTS) message to the node for which the data is destined. If a node wins the contention by choosing a unique time slot, the intended recipient of the RTS message replies with a Clear-To-Send (CTS) message. A node receiving the CTS reply is allowed to transmit its data packet. If a node contends unsuccessfully, CW is doubled (up to CW_{max}). CW is set to CW_{min} for each new packet. RTS and CTS messages contain information about the length of the intended transmission; this allows nodes that do not participate in the data transmission and that received either the RTS or CTS message to defer until that transmission is over [5, 6].

The CSMA/CA MAC protocol is efficient at low traffic loads since a node would be granted access to the medium almost immediately, resulting in a very short delay. However, the protocol is inefficient at high traffic loads, as collisions can occur frequently, obstructing data transmission.

We investigated this protocol since it is used in the highly popular IEEE 802.11 standard [7], and because of its potential improvement by optimizing the probability that nodes have a successful contention period while keeping the contention period as short as possible.

1.3 Literature Synopsis

1.3.1 Decision-Making Under Uncertainty

Decision-making under uncertainty is required to respond to system information (in our case, information about the network). Existing methods are inefficient and impractical (for problems with a significant number of variables), since there is typically an exponential growth in complexity with the number of variables. Currently there are 2 approaches to making decisions under uncertainty: Decision trees and strong junction trees.

1.3.1.1 Decision Trees

Decision trees represent a problem using a tree structure. The nodes in the tree represent variables, creating a branch for each outcome of the variable, which is connected to multiple copies of the next variable in time. This structure has the advantage of explicitly stating how variables affect one another in every scenario [1, 2]. However, this results in a large structure that is both difficult to set up and may contain a lot of duplications. The number of leaf nodes in decision trees grows exponentially with the number of variables. Thus decision trees quickly become impractical even with a few variables.

Decision trees always make an assumption known as the perfect recall assumption -- a decision always knows the result of all previous decisions [1, 2]. This assumption is not always valid or practical. The advantage of this method is that once the decision tree is set up, it is easy to compute decision strategies, which will always result in the global optimal solution.

Regardless of the advantages, the perfect recall assumption, the size of the structure, and the complexity of creating a decision tree make it impractical in most problems with a significant number of variables.

1.3.1.2 Strong Junction Trees

With the help of decision theory, JTs have been expanded in order to accommodate utilities and decisions [1, 2, 3]. The expansion defines a decision potential as a combination of a probability and utility factors.

Current theory state that in order to use a JT for solving a decision-making problem, a *strong* JT has to be used. A *strong* JT is a JT where the structure of the graph is determined by using the variable elimination technique with a fixed order; variables are to be eliminated in reverse chronological order. Note that variables grouped in the same time frame (the chronological order between these variables cannot be determined) are allowed to be ordered in any way. This further increases the size of the required tables. Since current theory has not yet defined the normalization operation for these decision potentials, the more general cluster graph method cannot be used.

The strong JTs are impractical in most problems with a significant number of variables, since they typically require a significant amount of memory. We expand this theory by defining the normalization operation, allowing for (loopy) decision cluster graphs.

1.3.2 Existing Adaptive MAC Protocols

While a very limited number of other adaptive MAC protocols exist, none of them implement PGMs.

[8] focuses on a specific use case for the CSMA/CA MAC protocol, namely the case where all nodes transmit data (with a small load) to a central node while this node transmits data (with a bigger load) to every other node. The length of the contention window for the central node is heuristically determined based on the number of packets it has sent and received.

[9] allows nodes implementing the CSMA/CA MAC protocol to estimate the number of transmitting nodes heuristically by observing the number of contention slots for which the channel is busy. The nodes use this estimation to adapt their contention windows.

However, neither infer the probabilities of the relevant random variables to create an adaptive protocol; they use heuristic methods and little information to create an adaptive protocol. The advantage of inferring the probabilities of random variables is that a variety of conditions and properties of nodes and the environment can be estimated, such as the data rate of nodes and whether there is noise in the channel. These conditions and properties allow for better estimations and a more effective adaptive protocol.

1.4 Objectives

The objectives of this work are as follows:

- Implement an effective method for making decisions in MAC protocols under uncertainties regarding wireless networks. Current methods for decision-making under uncertainty are inefficient, and it is therefore necessary to create an effective method for solving the general decision-making problem. This can be achieved by expanding the current theory of strong JT (containing decisions) to allow for (loopy) decision-cluster graphs.
- Improve the performance (and demonstrate the feasibility) of augmenting MAC protocols (specifically for wireless networks) with PGMs:
  - Implement a PGM in the Round Robin MAC protocol to determine the polling order in order to decrease the transmission delay at low or unequal traffic loads.
  - Implement a PGM in the CSMA/CA (802.11) MAC protocol to determine the length of the contention window in order to have a short and successful contention period.

1.5 Summary of Results and Contributions

This work achieved the following:

- Created an effective method for making decisions under uncertainty by expanding the current theory of strong JT (containing decisions) to allow for (loopy) decision cluster graphs. However, similar to the behaviour of (loopy) cluster graphs (upon which it is based), this method also leads to imprecise probabilities and utilities, and sub-optimal decision strategies. Note that extensive testing of this method (for general decision-making problems) is beyond the scope of this project and is recommended for future work.
- Implement a novel method for creating adaptive MAC protocols by augmenting MAC protocols with PGMs:
  - Implement a (loopy) decision cluster graph in the Round Robin MAC protocol for determining the polling order dynamically, showing that a (loopy) decision cluster graph can be practical.

- Implement a strong JT in the Round Robin MAC protocol for determining the polling order dynamically, illustrating that the performance of the (loopy) decision cluster graph is similar to that of a strong JT.
- Implement a cluster graph (without decisions) in the Round Robin MAC protocol for determining the polling order dynamically, showing that in certain cases the decision-making problem can be solved without utilities, avoiding the issues of (loopy) decision cluster graphs.
- These PGM-augmented Round Robin MAC protocols maintain fairness in the network and improves performance by reducing latency when traffic loads are unequal; this shows that augmenting the Round Robin MAC protocol with a PGM is beneficial.
- Implement a cluster graph in the CSMA/CA MAC protocol, inferring the number of contending nodes and changing the length of the contention window accordingly, showing that an augmented CSMA/CA MAC protocol is feasible.
- The PGM-augmented CSMA/CA MAC protocol improves fairness and throughput by optimizing the contention period, showing that a PGM-augmented CSMA/CA MAC protocol is beneficial.
- Estimating uncertainties in networks with the help of PGMs (even a simple one) and implementing an adaptive protocol according to these uncertainties is feasible and can improve performance.

1.6 Overview

Our main objective is to augment MAC protocols in wireless networks with PGMs in order to improve performance; thus background information about PGMs and existing MAC protocols is required (Chapter 2).

An efficient method for making decisions under uncertainty is required in order to respond to information about the network. Current theory uses strong JTs to assign decision strategies, which is inefficient due to the fact that the size of decision potentials grows exponentially with the number of RVs. Thus we expand current theory and create an efficient method for making decisions under uncertainty (Chapter 3). Since loopy cluster graphs allow for a more efficient and practical approach to solving probabilities compared to JTs, we similarly expand the current theory of strong JTs to allow for loopy decision cluster graphs. This we achieve by defining the normalization operation for decision potentials (the other required operations are already defined). With all the required operations defined, loopy decision cluster graphs can now be created and used to calculate probabilities and utilities, and assign decision strategies. However, this method has the following flaws:

- An independence assumption between RVs can be made, which can be incorrect, leading to incorrect probabilities and utilities, and sub-optimal decision strategies.
- The method can lead to making decisions that do not rely on the input variables of the decision strategy. Observing the input variables before executing the method prevents this from happening. However, observing input variables of decisions that are made in the future is impossible. Thus we recommend assigning a temporary local optimal solution for those decision strategies and when the system needs to make those decisions, their input variables are observed and the method is executed again.
- As with strong JTs, decisions whose optimal strategies rely on each other do not give a global optimal solution if they are optimized one at a time. If possible, these decision strategies should be optimized together for global optimal strategies.

Extensive testing of decision cluster graphs is beyond the scope of this project. Nonetheless, we show that decision cluster graphs can be used to solve a practical decision-making problem by implementing one in the Round Robin MAC protocol, using a decision cluster graph to determine which node to poll next (Section 4.2). We also implement a strong JT for this problem, comparing the performance and showing that a decision cluster graph can give results similar to that of a strong JT.

We show, with a practical example, that certain decision-making problems can also be solved without utilities -- making a decision based on probabilities of specific RVs (calculated with a cluster graph), thus avoiding the above-mentioned issues of decision cluster graphs. We use this method to solve the same practical decision-making problem, deciding which node to poll in the Round Robin MAC protocol (Section 4.3).

In order to keep the models used for augmenting the Round Robin MAC protocol simple, the observed variables were discretized. This allows for easier logical reasoning but it destroys information, since similar values are regarded as equal. This result in slightly sub-optimal decisions.

Creating an adaptive CSMA/CA MAC protocol is accomplished by using a cluster graph to estimate the number of nodes contending for channel access and adapting the length of the contention window accordingly (Chapter 5). We show that a linear correlation exists between the number of contending nodes and the optimal length of the contention window. Thus it is not necessary to implement a decision cluster graph for this problem; the simplistic linear correlation is used to determine the size of the contention window extremely efficiently. The issues with discretizing observed variables (found in the Round Robin protocols) are avoided in this model by using functions to determine the probabilities of the RVs that depend on the observed variables.

We evaluated the performance of our augmented methods by simulating wireless networks and comparing the performance of our augmented protocols to the standard protocols (Chapter 6).

Compared with the average latency of the Round Robin MAC protocols for unequal traffic loads (Fig. 1.1), the PGM-augmented protocols have a significantly lower average latency for data packets. This improvement is possible since the PGM-augmented methods estimate which nodes have a higher data rate and poll them more often. By polling these nodes more often, there is a shorter waiting time for these nodes to transmit their data packet(s), reducing the latency of those packets. From this figure, it is also evident that the decision cluster graph method performs similarly to the strong JT method.

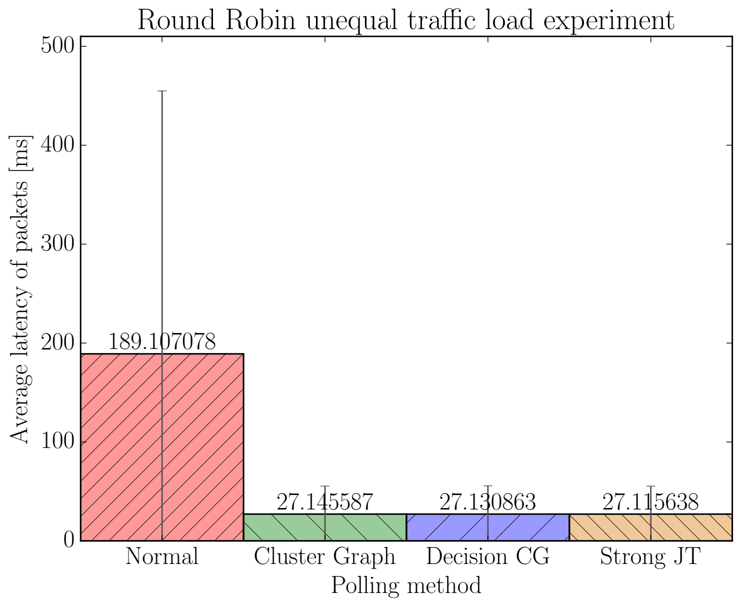


Figure 1.1: Average latency for the Round Robin unequal traffic load experiment. Note that the augmented Round Robin MAC protocols have a significantly lower average latency than the normal Round Robin MAC protocol.

Fig. 1.2 illustrates the efficient channel usage of the CSMA/CA MAC protocols for a saturated traffic load (the maximum stable traffic load). From this figure, we can conclude that the standard protocol performs excellently subject to the parameter choices and the number of nodes that participate in data transmissions. However, the PGM-augmented method performs similarly regardless of the number of nodes, showing that the PGM-augmented method is able to better adapt to network size and traffic loads by estimating the number of contending nodes and changing the size of the contention window accordingly.

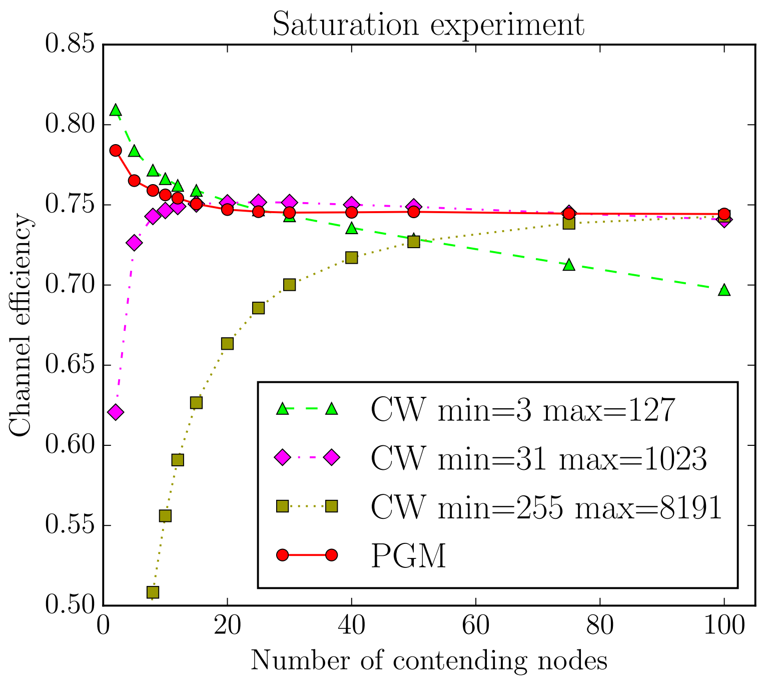


Figure 1.2: Channel efficiency for the CSMA/CA saturation test. Note that the PGM-augmented CSMA/CA MAC protocol maintains an efficient channel usage regardless of the number of contending nodes, while the efficiency of the normal CSMA/CA MAC protocol diminishes depending on the number of contending nodes and the parameter choices.

These results show that estimating uncertainties in a network with the help of a PGMs (even a simple one) and implementing an adaptive protocol according to these uncertainties is feasible and can improve performance.

Chapter 2

Theoretical Background

In this Chapter we provide only the theoretical background required for this work (Probabilistic Graphical Models (PGMs) and Medium Access Control (MAC) protocols for wireless networks). Since other literature concerning adaptive MAC protocols is limited, the full literature study can be found in Chapter 1. In the interest of brevity, we present only a brief overview of the MAC protocols and how the protocol could be augmented with a PGM. However, further detail regarding the protocols that we augment, i.e. the Round Robin and CSMA/CA MAC protocols, can be found in Chapter 4 and 5 respectively.

2.1 Probabilistic Graphical Models

Probabilistic Graphical Models have proven to be a powerful and effective tool for predicting the behaviour of probabilistic systems and assisting with their subsequent optimisation. Their applicability to wireless networks seems promising, since there are uncertainties in the network. In this work we specifically use Bayes networks (a PGM) for illustrating the dependencies between random variables (RVs), and we use cluster graphs (another PGM) to calculate the probabilities of these RVs.

2.1.1 Bayes Networks

A Bayes network is a directed, acyclic graph, illustrating the dependencies of RVs. Bayes networks can be used to calculate probabilities (for example, in the variable elimination method [1, 2]). However, for this work, we use Bayes networks to illustrate dependencies and to determine statistical dependencies between RVs. RV X_1 and X_n are statistically dependent if an active path exists between them [1, 2]. An active path is any trail, $X_1 \leftrightarrow ... \leftrightarrow X_n$, between X_1 and X_n, that satisfies the following conditions:

- Whenever there is a v-structure, $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, in the trail, then X_i or one of its descendants must be observed.

- No other node on the trail is observed.

2.1.2 Cluster Graphs

In this work we use cluster graphs to calculate the probabilities of the relevant RVs. Cluster graphs are capable of calculating the probabilities of RVs efficiently by exploiting Bayes' rule [1, 2].

Cluster graphs work as follows:

- Probability information about RVs is stored in tables (called probability factors). These probability factors form cluster nodes in the graph. Cluster nodes are denoted with ovals and labelled with the corresponding RVs in the graph.

- These cluster nodes are connected to one another with separation sets (*sep-sets*), the overlapping RVs of the two cluster nodes. Cluster nodes that have no mutual RVs are not connected. These connections must comply with the running intersection property: any 2 cluster nodes containing variable X is connected with a unique path of sep-sets and other cluster nodes containing X. Sep-sets are denoted with rectangles and labelled with the corresponding RVs in the graph.

- Observed RVs are variables whose states are known, thus the effects they have on the remaining variables are calculated and stored, at which point the observed RVs are removed from the cluster graph. This is achieved by modifying the probability factor table (which contains the observed RV) as follows: entries corresponding to the observed value of the RV are kept, while remaining entries are discarded.

- Messages (a probability factor containing the RVs of the sep-set) are sent between the cluster nodes (known as belief propagation). These messages are propagated until their values converge. Note that convergence is not guaranteed.

- Sending the messages requires the following operations on probability factors to be defined:

  - Absorb -- absorbing 2 (or more) probability factors into a combined probability factor:
  $$\rho_1 \oplus \rho_2 = \rho_1 \cdot \rho_2$$

  - Cancel -- cancelling (or dividing) a probability factor out of another probability factor:
  $$\rho_1 \ominus \rho_2 = \rho_1 / \rho_2$$
  - Marginalize -- marginalizing a probability factor to contain only a subset of the RVs:
  $$marg_W(\rho) = \sum_{W'} \rho$$
  - Normalize -- normalizing the values in the probability factor:
  $$norm(\rho) = \frac{\rho}{\sum \rho}$$

- There are 2 methods of belief propagation, each with its own method for calculating messages:

  - Loopy belief propagation: a message from cluster node A, B, C towards cluster node B, C, D ($\rho_{ABC-BCD}$, a probability factor), with sep-set B, C, is calculated as follows:
    1. Absorb the probability factor of cluster node A, B, C with all other incoming messages to cluster node A, B, C.
    2. Marginalize the resulting factor to contain only B, C.
    3. Normalize the probability factor.

    In this method, the probability factors of the cluster nodes remain constant, while the probability factors of the messages are updated.
  - Loopy belief update: message $\rho_{ABC-BCD}$, with sep-set B, C, is calculated as follows:
    1. Cancel message $\rho_{BCD-ABC}$ from the probability factor of cluster node A, B, C.
    2. Marginalize the resulting factor to contain only B, C.
    3. Normalize the probability factor.

    However, in this method, the probability factors of both the messages and cluster nodes are updated. The probability factors of cluster nodes are updated if any of the incoming messages change -- the new probability factor of a cluster node equals the original probability factor of a cluster node, absorbed by all incoming messages.

- Once the messages converge, the probabilities of RVs can be calculated. The probabilities are calculated depending on the belief-propagation method:

  - Loopy belief propagation: The probability factor of a cluster node containing the queried RVs and all incoming messages towards the cluster node is absorbed and marginalized to contain only the queried RVs.

  – Loopy belief update: The probability factor of a cluster node containing the queried RVs is marginalized to contain only the queried RVs.

Note that the probabilities are not guaranteed to be exact, as it can converge to incorrect probabilities.

If the cluster nodes are connected in such a way that no loops are formed, the cluster graph is also known as a junction tree (JT). Junction trees result in exact calculations and always converge with one set of message passing. Since messages are only sent once, the normalization operation is not strictly necessary. However, JTs typically have more RVs in each cluster node, which result in an exponential growth in the size of the table with each extra RV. This exponential growth makes JTs impractical for most problems with a significant nummber of RVs. Even though loopy cluster graphs are imprecise, the results they give are typically close enough to the correct solution. In addition, the significantly reduced size of the probability factors makes them far more practical. For an in-depth explanation of cluster graphs, refer to [1, 2].

2.2 Medium Access Control Protocols in Wireless Networks

In this section we consider the different MAC protocols used in wireless networks. A brief overview of the protocol is given, explaining its advantages and disadvantages. We also suggest how the protocol could be improved with a PGM. Typically, wireless MAC protocols are divided in two main groups, namely deterministic and contention MAC protocols.

2.2.1 Deterministic MAC Protocols

In the case of deterministic MAC protocols, nodes in the network have a fixed order in which they are allowed to transmit data. These protocols typically have the following properties:

- The medium is collision free (2 or more nodes whose transmissions would interfere with one another never transmit at the same time).
- The protocols are efficient at high traffic loads but have a transmission delay at low traffic loads.
- Nodes should typically be in transmission range of either a master node or all the nodes in the network.

2.2.1.1 Round Robin Protocol

The Round Robin protocol works as follows: A node is appointed as the master node, which polls the other (slave) nodes in the network in sequence. The polled node is allowed to transmit its data [5].

Advantages:

- The medium is collision free, as the master node controls access to the medium.
- It is efficient at high traffic loads, since there are no collisions.
- The failure of a slave node does not have a significant impact on the network, as the master node will simply poll the next node.

Disadvantages:

- It is inefficient at low (or unequal) traffic loads due to polling overhead; a node with data to transmit has to wait to be polled, increasing the transmission delay (latency) of data packets.
- If the master node fails, the network fails.

This protocol could be optimized by means of an unfixed polling sequence, allowing for the polling of the node with the highest probability of having data to transmit. This should improve performance at low (or unequal) traffic loads, while maintaining the performance at high traffic loads. We augment the Round Robin MAC protocol in Chapter 4, since it is a simple deterministic MAC protocol and other deterministic MAC protocols offer similar performance, making it an ideal test-case protocol.

2.2.1.2 Binary Tree Protocol

This is a limited-contention protocol that works as follows: All nodes are divided into groups and each node is assigned to more than one group. In fact, the groups are the nodes of a balanced tree, and the nodes are the leaves of the tree. A node is part of all the groups of which it is a descendant. Starting with the group at the top of the tree as the current group, all nodes of the current group are allowed to transmit data. If a collision occurs, the next group down the tree becomes the current group (a smaller number of nodes, thus with a greater chance of success). This occurs until all the nodes waiting to transmit are successful and then the process repeats [5].

Advantages:

- It is efficient at low traffic loads as a node is granted channel access quickly, resulting in a short delay.

- Node failure does not have a big impact on the network.
- It is fairly efficient at high traffic loads, as there is little extra overhead thanks to the limited contention.

Disadvantages:

- All nodes need to be within transmission range of all other nodes.
- For contention to remain limited, the nodes should be synchronized in respect of which group can contend for the channel.
- Adding nodes to the network is slightly more complicated, since it is preferred to have a balanced tree.

A possible improvement is to allow nodes to start with a group lower down the tree, since this would improve the performance under high traffic loads (less contention overhead). The problem with this approach is that if all the nodes do not start with the same group, contention is no longer limited, increasing the number of collisions significantly and degrading the performance of the network.

2.2.1.3 Slot Reservation Protocol

Also named Bit-Map or Binary countdown protocol. Nodes using this protocol are assigned a 1-bit slot in the slot period. A node transmits a 1 in its slot to indicate that it has data to transmit. After the slot period, the nodes transmit according to the indicated slots [5].

Advantages:

- The protocol is collision free, as each node has its own slot.
- It is efficient at high traffic loads, because there are no collisions.
- Node failure has a small impact on performance, resulting in an empty slot.

Disadvantages:

- At low traffic loads with a large number of nodes in the network, the length of the slot period can become long relative to the length of the data transfer period, resulting in an inefficient use of the channel.
- All nodes need to be within transmission range of all other nodes in order to know if a slot is reserved.
- A 100% success rate during the slot period is assumed, which cannot be guaranteed.

This protocol does not leave room for improvement.

2.2.1.4 Token Bus Protocol

In this protocol, a virtual token is created, which is transmitted from node to node (in a fixed order). If a node possesses the token, it has permission to transmit data [5].

Advantages:

- Only one node (the one with the token) is allowed to transmit, resulting in a collision-free protocol.
- It is efficient at high traffic loads thanks to near-constant data transmissions and the absence of collisions.

Disadvantages:

- Node failure can have an impact on the network, since it could result in 'losing' the token and breaking the protocol.
- The latency of data packets at low (or unequal) traffic loads could become long, since the node has to wait until it possesses the token.

An adaptive token-passing order makes it possible to improve this protocol during low traffic loads. This is achieved by transmitting the token to the node with the highest probability of having data to transmit.

2.2.2 Contention MAC Protocols

Contention MAC protocols require nodes to contend for access to the channel. These protocols typically have the following properties:

- They are inefficient at high traffic loads due to frequent collisions in the channel while nodes contend for channel access.
- They are efficient at low traffic loads, since the contention period is short and successful with a small number of contending nodes.
- Nodes have to be within transmission range of the destination node only.

2.2.2.1 ALOHA Protocol

ALOHA allows a node to transmit data immediately (a variation of this protocol divides time into slots, allowing transmissions at the start of slots only). The node then waits for an acknowledgement message from the destination. If it does not receive said message, the node waits a random time length and repeats the process [6].

Advantages:

- An extremely short delay is possible during low traffic loads, as the node is permitted to start transmitting immediately.
- The node needs to be within transmission range of the destination node only.

Disadvantages:

- There are frequent collisions (especially during high traffic loads) as the channel activity is not checked; nodes simply start transmitting.

Having a smart or adaptive back-off period (the wait time after an unsuccessful attempt) could reduce collisions at high traffic loads, while maintaining a short delay at low loads.

2.2.2.2 Carrier Sense Multiple Access (CSMA) Protocol

In this protocol, the node listens to the channel until it is idle for a short period. When this criterion is met, the node backs off a random number of idle time slots, transmits its data and waits for an acknowledgement message. If the acknowledgement message is not received shortly, it doubles the back-off time and repeats the process [5].

Advantages:

- The delay at low traffic load is relatively small; a node can start transmitting shortly after the channel becomes idle.
- The node needs to be within transmission range of the destination node only.

Disadvantages:

- During high traffic loads, collisions occur frequently, resulting in extended back-off periods and inefficient usage of the channel.

Improving this protocol is possible by means of an adaptive back-off period. A short back-off period allows for short delays at low traffic loads, while a long back-off period improves the probability of a successful contention, minimizing collisions. We augment the CSMA MAC protocol in Chapter 5, as it is used in the highly popular IEEE 802.11 standard.

Chapter 3

Decisions in Cluster Graphs

Allowing a network protocol to be more dynamic can improve its performance, since it can react to a wider variety of changes in the network. This can be accomplished by allowing the protocol to make calculated decisions. However, the decision-making task is complex and an efficient approach is required to solve this problem. Currently, the most efficient method for making decisions under uncertainty is the use of a strong junction tree (JT). Strong JTs are inefficient, thus we expand the theory to allow for (loopy) *decision cluster graphs*, reducing the memory requirements significantly. Similar to normal (loopy) cluster graphs, results for loopy decision cluster graphs are not guaranteed to be exact, resulting in sub-optimal decision strategies.

3.1 Theoretical Background

3.1.1 Decision Theory

Responding to system information implies decision-making, and since there is uncertainty in wireless networks, decision-making under uncertainty is necessary. The decision theory framework provides a foundation for the decision-making task. Decisions are represented as decision variables for which the user can assign probabilities (also know as a decision strategy). A decision can rely on both random variables (RVs) and other decision variables; in these cases the decision strategy would specify which action to take given the input variables [1, 2, 3].

In order to make decisions, the system requires information about how preferable the outcomes of certain variables are. This is achieved by assigning numerical values (also known as utilities) to the outcomes of certain variables (this information can be stored in a table known as a utility factor). These assignments allow us to define an expected utility, stating how preferable a situation is when there is uncertainty regarding the relevant RVs. Decision are made in order to maximize the expected utility, creating the most preferred situation [2, 3, 4].

3.1.2 Bayes Networks and Influence Diagrams

Bayes networks are used to give a graphical illustration of dependencies between RVs, and to determine if RVs are statistically dependent. RVs X_1 and X_n are statistically dependent if there is an active path between them. An active path is any trail of connected RVs, $X_1 \leftrightarrow ... \leftrightarrow X_n$, between X_1 and X_n that comply with the following [1]:

- Whenever there is a v-structure, $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ in the trail, then X_i or one of its descendants must be observed.
- No other node on the trail is observed.

Bayes networks are very useful to illustrate the dependencies of random variables. However, Bayes networks do not include utilities and decisions. Influence diagrams extend Bayes networks by including decision and utility nodes [10, 11]. Random variables (denoted with ovals) may depend on the states of both random and decision variables. Decision variables (denoted with rectangles) are essentially random variables where the user can assign a strategy. The input variables of a decision strategy are the parents of the decision variable (which can be both random and decision variables). An edge into the decision variable is also known as an information edge. Utility nodes (denoted with diamonds) do not add variables, but assign utilities to certain outcomes [11]. Note that if an influence diagram is to be used to determine statistical dependence between variables, the trail of the active path cannot pass through utility nodes, as utility nodes do not imply a statistical dependence -- they only describe how preferable the outcomes of the variables are.

Influence diagrams are quite general and versatile as they do not necessarily make the perfect recall assumption (a decision strategy that depends on all previous decisions and their parents). In this work we use limited memory influence diagrams (LIMIDs): influence diagrams that do not make the perfect recall assumption, representing all information edges explicitly [10]. Not making the perfect recall assumption enables us to model bigger problems, since making the perfect recall assumption results in an exponential growth in the size of decision strategies with each new decision. However, it results in strategies prone to a local-optimal solution. Fig. 3.1 illustrates an example of a LIMID, illustrating all information edges explicitly.

3.1.3 Relevance Graphs

Assigning a decision strategy for one decision variable can cause a (previously optimal) decision strategy for another decision variable to become suboptimal. A

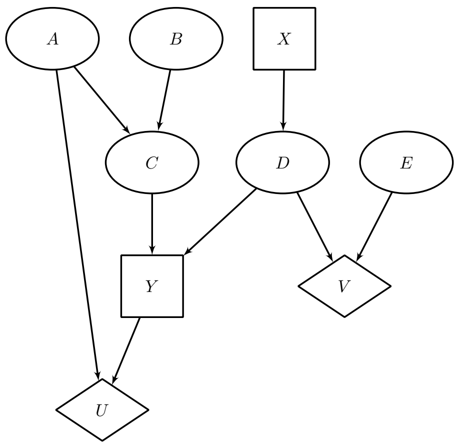


Figure 3.1: Example LIMID for illustration purposes. Random variables: A,B,C,D,E. Decision variables: X,Y. Utility nodes: U,V.

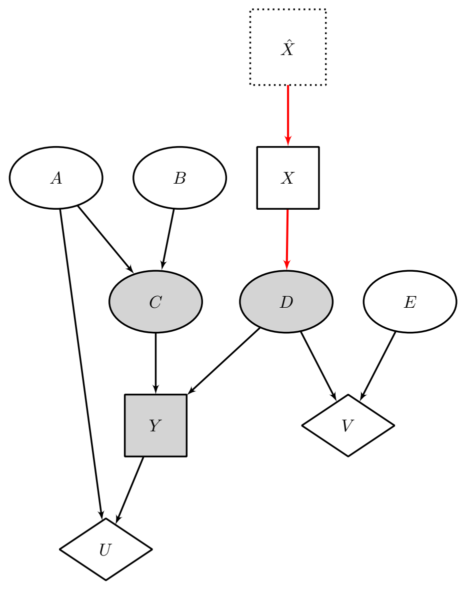
relevance graph is a graphical structure that displays these strategic dependencies between different decision variables [10].

To determine which decision variables are strategically relevant to which, a graphical criterion called S-reachability is used. A decision variable X is S-reachable from a decision variable Y (Y relies on X) in an influence diagram if there is some utility node U, where U is a descendant of Y, such that if a new parent $\widehat{X}$ were added to X, there would be an active path (see Section 2.1.1) from $\widehat{X}$ to U given $Family_Y$ (node Y and its immediate parents) [10].

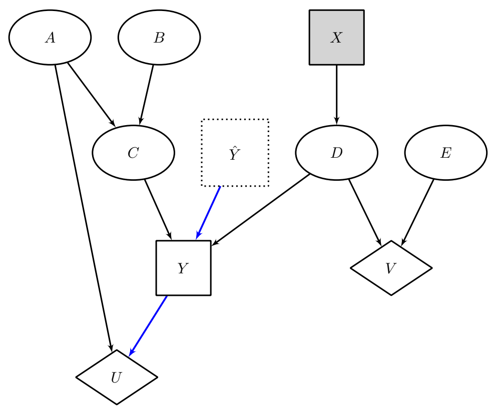
Fig. 3.2 illustrates S-reachability and the relevance graph of the influence diagram in Fig. 3.1.

A relevance graph is a directed (possibly cyclic) graph containing the decision variables of an influence diagram. A directed edge $X \rightarrow Y$ exists if X is strategically relevant to Y (X is S-reachable from Y) [10].

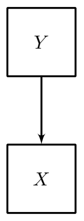
In an acyclic relevance graph, decisions are optimized according to the topological ordering of the decision variables in the graph (for every directed edge XY from vertex X to vertex Y, X comes before Y in the ordering). Cyclic relevance graphs are common in LIMIDs and do not have a fixed order for optimizing decisions, resulting in local optimal solutions [10].



(a) S-reachability for X from Y, used to determine if the $X \to Y$ edge exists in the relevance graph. Since there is no active path from $\widehat{X}$ to a utility node, the edge does not exist.



(b) S-reachability for Y from X, used to determine if the $Y \to X$ edge exists in the relevance graph. Since there is an active path from $\widehat{Y}$ to utility node U, the edge does exist.



(c) Resulting relevance graph; created from the results of the S-reachability for X and Y.

Figure 3.2: Setting up a relevance graph for our example LIMID (Fig. 3.1). Fig. 3.2a illustrates the S-reachability for X from Y and Fig. 3.2b illustrates the S-reachability for Y from X. Fig. 3.2c illustrates the resulting relevance graph.

3.1.4 Decision Potentials

Probability factors are the basic data structures used in cluster graphs, but no utility information is stored in them. Thus probability factors have been extended to decision potentials, which contain both probability and utility information [3, 10, 11].

A decision potential structure is represented as follows:

$$\begin{aligned}&\gamma = (\rho, \mu)\\ &\rho : probability\ factor\\ &\mu : utility\ factor\end{aligned}$$

Extending factors to decision potentials is accomplished as follows:

1. Probability factors: $\mu = 0$, example illustrated in Table 3.1
2. Unassigned decision strategies: $\rho = 1, \mu = 0$, example illustrated in Table 3.2
3. Utility factors: $\rho = 1$, example illustrated in Table 3.3

$A\ B\ C$	$\rho(C\|A,B)$	$\mu(C\|A,B)$
0 0 0	0.8	0
0 0 1	0.2	0
0 1 0	0.6	0
0 1 1	0.4	0
1 0 0	0.1	0
1 0 1	0.9	0
1 1 0	0.25	0
1 1 1	0.75	0

Table 3.1: Example decision potential $\gamma(C|A,B)$ for probability factor $\rho(C|A,B)$ of Fig. 3.1. Note that $\mu = 0$.

$C\ D\ Y$	$\rho(Y\|C,D)$	$\mu(Y\|C,D)$
0 0 0	1	0
0 0 1	1	0
0 1 0	1	0
0 1 1	1	0
1 0 0	1	0
1 0 1	1	0
1 1 0	1	0
1 1 1	1	0

Table 3.2: Example decision potential $\gamma(Y|C,D)$ for an unassigned decision strategy factor $\rho(Y|C,D)$ of Fig. 3.1. Note that $\rho = 1$ and $\mu = 0$.

A Y	$\rho(A,Y)$	$\mu(A,Y)$
0 0	1	-10
0 1	1	50
1 0	1	20
1 1	1	32.5

Table 3.3: Example decision potential $\gamma(A,Y)$ for utility factor $\mu(A,Y)$, node U of Fig. 3.1. Note that $\rho = 1$.

3.2 Decision Potentials Operations

Current theory uses strong JTs to assign decision strategies. However, strong JTs are impractical in most problems, since the size of the decision potentials grows exponentially in the process of forming a JT. By limiting the order in which variables are eliminated to form the JT (and thus a *strong* JT), the size of the decision potentials grows even further.

Similarly to cluster graphs with only probabilities, expanding decision potentials to allow for (loopy) decision cluster graphs will reduce memory requirements significantly and create a more practical approach to assigning decision strategies. However, decision cluster graphs require that a few operations be defined. The theory used for strong JTs [3, 10, 11] defines all the required operations except the normalization operation, since normalization is not strictly required for strong JTs. The operations are listed below:

Absorb

$$\gamma_1 \oplus \gamma_2 = (\rho_1 \cdot \rho_2 \ , \ \mu_1 + \mu_2) \tag{3.1}$$

Cancel

$$\gamma_1 \ominus \gamma_2 = (\rho_1 / \rho_2 \ , \ \mu_1 - \mu_2) \tag{3.2}$$

Marginalize

$$marg_W(\gamma) = \left(\sum_{W'} \rho \ , \ \frac{\sum_{W'} \rho \cdot \mu}{\sum_{W'} \rho} \right) \tag{3.3}$$

Normalize

We suggest normalizing as follows:

$$norm(\gamma) = \left(\frac{\rho}{\sum \rho} \ , \ \mu - \frac{\sum \rho \cdot \mu}{\sum \rho} \right) \tag{3.4}$$

3.2.1 Normalization of Decision Potentials

Normalization is important when the cluster graph is loopy, since it ensures that all the potentials are on the same scale. We are concerned with the normalization of utilities, since normalization of probabilities is already defined. Normalization should have the following properties:

$$norm(\gamma) = norm(norm(\gamma)) \tag{3.5}$$

$$norm(\gamma_1 \oplus \gamma_2) = norm(\gamma_1) \oplus norm(\gamma_2) \tag{3.6}$$

There are 3 general types of strategies we can attempt in order to normalize utilities:

1. *Do nothing.* This method works for JTs, since it does not change the scale of utilities at all. It also satisfies both equation (3.5) and (3.6). However, in loopy structures, it does not work at all. When the absorb function is used in a loop without normalization, the utilities will continue to grow as utilities are added together. This creates two problems: Utilities will never converge, and the RVs inside the loop will become ever more important as their utilities grow (in either the positive or negative direction).

2. *Scaling.* Using a local scaling function (i.e. using only the utility values from the local decision potential). For example, we scale to let utilities fall in a fixed range $(-1, 1)$. This would cause smaller utilities to become more important and larger utilities less important, completely disregarding the size and impact of utilities. A global scaling factor is more reasonable, since the relationship between different utilities nodes is kept. However, this is an expensive normalization method, since for each normalization, all the nodes in the cluster graph must be considered.

3. *Shifting* (add/subtract). Assignment of decision strategies is dependent on the utilities only, and shifting all the utilities in a decision potential therefore does not affect the decision strategy (assigning decisions strategies is covered in Section 3.4). Also, normalizing usually corresponds to marginalization as follows:

$$norm(\gamma) = \gamma \ominus marg_{\emptyset}(\gamma)$$

Applying this results in the following normalization operation:

$$norm(\gamma) = \left(\frac{\rho}{\sum \rho} \; , \; \mu - \frac{\sum \rho \cdot \mu}{\sum \rho} \right)$$

This normalization operation satisfies both equation (3.5) and (3.6) . In the interest of brevity, we relegate our proof of this to Appendix A. The addition of this operation makes it possible to apply belief propagation techniques to loopy decision cluster graphs, something that (to the best of our knowledge) was not possible before.

3.3 Belief Propagation in Decision Cluster Graphs

Similarly to normal cluster graphs, decision cluster graphs perform belief-propagation to calculate probabilities and utilities. However, before belief propagation can be executed, the decision cluster graph has to be created. Creating a decision cluster graph is exactly the same as normal cluster graphs, using decision potentials as the cluster nodes and connecting the cluster nodes with separation sets conforming to the running intersection property.

With the created decision cluster graph and the required decision potential operations, both the loopy-belief-propagation and the loopy-belief-update algorithms can be used for calculating probabilities and utilities. However, we prefer the loopy-belief-propagation algorithm as it is simpler to update decision strategies (a simple replacement of decision potentials).

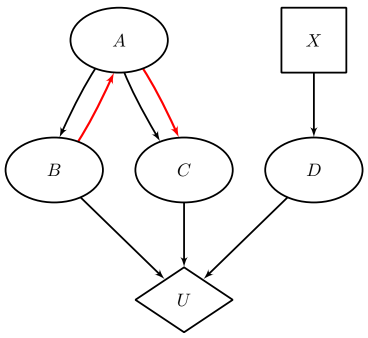
As with normal cluster graphs, the resulting probabilities and utilities are not guaranteed to be exact. Belief-propagation in loopy decision cluster graphs can also make an independence assumption, which results in inaccurate utilities. We illustrate with the help of an example: consider the influence diagram in Fig. 3.3a and the corresponding cluster graph in Fig. 3.3b.

Consider the message γ_{BCD-DX} and how it is calculated:

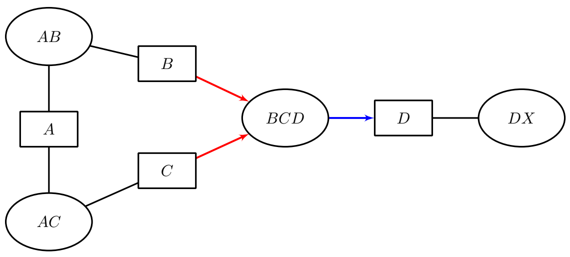
$$\gamma_{BCD-DX} = norm\left(marg_D\left(\gamma_{BCD} \oplus \gamma_{AB-BCD} \oplus \gamma_{AC-BCD}\right)\right) \tag{3.7}$$

$$\therefore \; \rho_{BCD-DX} = norm\left(marg_D\left(\rho_{BCD} \cdot \rho_{AB-BCD} \cdot \rho_{AC-BCD}\right)\right) \tag{3.8}$$

Since the cluster BCD contains no probability information (it is a utility cluster) and B & C are in separate separation sets, the assumption $\rho(B, C) = \rho(B) \cdot \rho(C)$ is made in this calculation (equation (3.8)). This is a poor assumption to make, since B and C are statistically dependent (illustrated with the active path in Figure 3.3a), and lead to inaccurate probabilities. When marginalizing the decision potential to calculate the message γ_{BCD-DX}, these inaccurate probabilities can cause the utilities to become highly inaccurate, since the marginalization of utilities is dependent on the probabilities (equation (3.3). These inaccurate utilities can change the decision strategy of X dramatically, since a decision strategy is made based on the utilities.



(a) Influence diagram for our example problem. Note that B and C are statistically dependent, as shown with the active path between B and C.



(b) Corresponding decision cluster graph for the influence diagram in Figure 3.3a, illustrating the required messages (γ_{AB-BCD} and γ_{AC-BCD}) to calculate the message γ_{BCD-DX}.

Figure 3.3: Graphical models illustrating how an independence assumption is made in loopy decision cluster graphs.

3.4 Assigning Decision Strategies

The objective of the decision-making task is to assign a decision strategy. Currently the method for assigning decision strategies in strong JT is as follow:

1. Initialize a default decision strategy by assigning a probability of 1 to all the entries in the decision potential of the decision strategy (as previously seen in Table 3.2).

2. Propagate all messages from the leaves of the tree towards the node containing the decision strategy.

3. Calculate the cluster belief of the node containing the decision strategy by absorbing the decision potential of the node with all incoming messages (similar to calculating the probabilities in a normal JT).

4. Marginalize the resulting decision potential to contain only the decision variable and all its input variables (all the variables present in the decision strategy).

5. For each assignment of the input variable(s), assign a probability of 1 to the value of the decision variable with the biggest utility value and 0 to the rest (updating the decision strategy potential).

This will maximize the overall expected utility [3, 10, 11].

Assigning decision strategies in (loopy) decision cluster graphs is similar, the exception being that before the cluster belief at the node containing the decision strategy can be calculated, belief-propagation has to be performed.

Unfortunately, assigning a decision strategy in a decision cluster graph is fundamentally flawed; the assigned decision strategy is only dependent on the variables in the separation set containing the decision variable (note that the set-up of a strong JT does not allow this to happen). In the worst case, the decision variable takes the same value regardless of the value of the input variable(s).

We illustrate this with the help of an example. In this example, the decision strategy that needs to be assigned is $\rho(D|R)$. Therefore we need to assign a probability of 1 to the value of decision variable D with the biggest utility value for each value of R. Consider the relevant part of a cluster graph (illustrated in Fig. 3.4), the decision potentials of the messages (Table 3.4 and Table 3.5) and the cluster belief (the decision potential in Table 3.6).

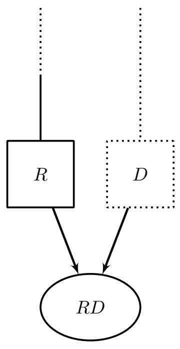


Figure 3.4: Part of a decision cluster graph (D = decision variable) used to calculate the cluster belief of node RD. The arrows indicate the messages. Cluster RD contains only the decision strategy $\gamma(D|R)$.

R	$\rho(R)$	$\mu(R)$
0	ρ_{R0}	μ_{R0}
1	ρ_{R1}	μ_{R1}

Table 3.4: Decision potential for message $\gamma(R)$, used to calculate the cluster belief of cluster RD.

D	$\rho(D)$	$\mu(D)$
0	ρ_{D0}	μ_{D0}
1	ρ_{D1}	μ_{D1}

Table 3.5: Decision potential for message $\gamma(D)$, used to calculate the cluster belief of cluster RD.

R D	$\rho(R, D)$	$\mu(R, D)$
0 0	$1 \cdot \rho_{R0} \cdot \rho_{D0}$	$0 + \mu_{R0} + \mu_{D0}$
0 1	$1 \cdot \rho_{R0} \cdot \rho_{D1}$	$0 + \mu_{R0} + \mu_{D1}$
1 0	$1 \cdot \rho_{R1} \cdot \rho_{D0}$	$0 + \mu_{R1} + \mu_{D0}$
1 1	$1 \cdot \rho_{R1} \cdot \rho_{D1}$	$0 + \mu_{R1} + \mu_{D1}$

Table 3.6: Cluster belief at decision node RD (illustrated in Fig. 3.4). Calculated by absorbing the decision potential of the default decision strategy with the incoming messages (message $\gamma(R)$, Table 3.4 and message $\gamma(D)$, Table 3.5).

From Table 3.6, the decision strategy for the different values of R would be calculated as follows:

$$R = 0 : (\mu_{R0} + \mu_{D0}) \ vs. \ (\mu_{R0} + \mu_{D1}) \tag{3.9}$$

$$(\mu_{D0}) \ vs. \ (\mu_{D1}) \tag{3.10}$$

$$R = 1 : (\mu_{R1} + \mu_{D0}) \ vs. \ (\mu_{R1} + \mu_{D1}) \tag{3.11}$$

$$(\mu_{D0}) \ vs. \ (\mu_{D1}) \tag{3.12}$$

Note that equation (3.10) = (3.12), thus the decision strategy is only dependent on μ_{D0} and μ_{D1}; the value of R is irrelevant.

This flaw can be partially avoided by observing the parents of the current decision variable (the decision that has to be made at this point in time) before performing belief-propagation. By definition, these RVs will be known at the time of the decision. This effectively removes these input variables of the decision strategy, and thus the optimal decision can be made. For future decisions this is not possible, and we suggest assigning a temporary strategy (which is re-assigned when that decision is the current decision -- time has elapsed, making the next decision in time the current decision). This results in a local optimal solution for future decisions, but the current decision strategy might be part of the global optimal solution. Another disadvantage of this strategy is that belief-propagation has to be performed when each decision is made. Thus a complete set of decision strategies cannot be determined beforehand. The effect of this flaw could also be minimized by structuring the problem in such a way that decision variables do not have input variables. However, the input variables to the system should still be observed and belief-propagation still needs to be performed for each decision in time.

Another method for avoiding this flaw is to ensure that a decision variable and its input variables are always in the same cluster and separation set. This can be achieved by absorbing the relevant decision potentials together. However, this increases the size of the decision potentials dramatically and the resulting cluster graph could be a JT.

3.5 Algorithm

We summarize the process of assigning decision strategies with the help of Algorithm 1:

Algorithm 1 Assigning Decision Strategies Using Decision Cluster Graphs

```
 1: Define RVs and decision variables.
 2: Define decision potentials of the problem (using default decision strategies for
    decision variables).
 3: Set up a LIMID for the problem.
 4: Create a relevance graph from the LIMID.
 5: Determine the sequence in which the decisions should be optimized using the
    relevance graph.
 6: Create a cluster graph from the decision potentials.
 7: repeat
 8:   Observe input variables of current decision variable.
 9:   while Current decision is not optimized do
10:     Perform belief propagation.
11:     Optimize the next decision strategy (according to the determined sequence):

12:       Calculate cluster belief at decision node.
13:       Update the decision potential of decision variable to optimal strategy.
14:   end while
15:   Current decision is made (decision variable is observed) and time progresses
      to next time instance (and a new current decision).
16:   Reset all remaining decisions to default strategies.
17: until All decisions are optimized
```

3.6 Conclusions and Recommendations

Using a decision cluster graph to assign decision strategies is efficient, since the size of the decision potentials is kept at a minimum. However, similar to the behaviour of (loopy) cluster graphs for calculating probabilities (upon which this method is based), this method also leads to inaccurate probabilities and utilities, and sup-optimal decision strategies. For problems where obtaining a global optimal solution is impractical, this method could prove effective. However, testing this method for general decision-making problems is beyond the scope of the project and we recommend it for future work.

The issues (and recommendations) we identified with (loopy) decision cluster graphs are as follows:

1. The independence assumption that is made (Section 3.3) can lead to inaccurate probabilities and utilities, resulting in sup-optimal decision strategies.

2. Decision strategies may not rely on the input variables of the decision variable (Section 3.4). This effect can be minimized by observing the input variables of the system. However, this requires that the method should be performed at each decision in time.

3. Cyclic relevance graphs result in sub-optimal decision strategies when the decision strategies are optimized one at a time. The reason for this is that the optimal strategies are reliant on each other and should, if possible, be optimized together for global optimal strategies.

In the next chapters we show that this method is viable, even with these flaws, by implementing it in a practical problem, using it to decide which node to poll in the Round Robin MAC protocol (Section 4.2). In order to compare the results of this method, we implement a strong JT for determining which node to poll in the Round Robin MAC protocol. We also show that certain decision-making problems can be solved without utilities, by making decisions based only on probabilities. We illustrate and explain this method by implementing it in the same problem, determining which node to poll in the Round Robin MAC protocol (Section 4.3).

Chapter 4

Implementing a PGM in the Round Robin MAC Protocol

The main objective of this work is to improve the performance (and thus demonstrate the feasibility) of augmenting MAC protocols with a PGM. We augment the Round Robin MAC protocol, since it is a simple deterministic MAC protocol that is easily augmented with a PGM by using a PGM to determine which node to poll next. In addition, other deterministic MAC protocols offer similar performance to the Round Robin MAC protocol, making it an ideal test-case protocol.

4.1 Objectives

The overall objective is to improve the performance of the Round Robin MAC protocol by augmenting it with a PGM. In the protocol, a master node polls the other (slave) nodes in the network to grant them access to the medium, allowing a node to transmit the data packets in its transmission queue (up to a maximum number of packets per poll) [5].

In the normal protocol, nodes are polled sequentially, making the protocol less efficient under low or unequal loads: a slave node has to wait for the other slave nodes (with empty queues) to be polled, wasting time and increasing the latency of other packets. However, this approach is efficient under high loads, since the master node controls the medium, preventing collisions in the medium.

An improved protocol can be created by changing the polling order to minimize polling nodes with empty queues. It is uncertain which node(s) has data packets to transmit, therefore we use a PGM to calculate the relevant probabilities. Our objective is to create a polling order for the slave nodes, which will minimize polling nodes with empty queues while maintaining a reasonably fair protocol. The performance improvement should be a reduced packet latency since nodes with data packets in their queue are granted access to the channel more quickly.

We approached this problem in two ways:

The first is based on decision theory, using our decision cluster graph method (Algorithm 1) to determine which node to poll (Section 4.2). In order to compare the performance of a decision cluster graph to current theory, we created the corresponding strong junction tree (JT). The second approach makes a decision based on probabilities only, using a normal cluster graph to calculate the probabilities (Section 4.3).

The second approach was performed to have a simpler and quicker method, and to illustrate another method for making decisions under uncertainty.

4.2 Round Robin Protocol Augmented with a Decision Cluster Graph

In this method we use a decision cluster graph in order to let the master node determine which slave node to poll next. Note that we also created the corresponding strong JT in order to compare the performance. However, before we can create the decision cluster graph, it is necessary to first identify the relevant variables of the problem.

4.2.1 Relevant Variables

There are a number of variables relevant to this problem, but first and foremost is the decision variable, deciding which slave node to poll (PID).

In order to improve the performance of the protocol, we want to avoid polling nodes that do not have data to transmit; thus we estimate the queue length for each node (CQ_i). Also, a node with a long queue can become congested, resulting in increased latencies for the packets.

We also want to ensure fairness in the network, thus we log the time elapsed since each node was polled (tsp_i) (if a node was polled long ago, it could indicate that the node is being treated unfairly). The time elapsed since a node was polled also influences performance, since it influences the latency of the packets (but only if there are packets in the node's queue).

The queue length of a node is dependant on this time elapsed since it was polled, but also on the data rate of the node (DR_i). We can estimate a node's data rate from the following observable variables: the previous number of packets that the node transmitted (pqs_i), and how long the node had to wait to be polled in order to transmit said packet(s) (pwt_i).

Finally, we are concerned with the new queue length (NQ_i) and the new time elapsed since a node was polled ($NTSP_i$) for after the current polling has taken place.

We use a variable NP_i (a binary RV) for each node, indicating if node i is the node to be polled.

We use discrete variables, as the calculations are much simpler and it is easier to create logical functions between the variables. Although the observed variables are actually continuous, we discretize them in three levels in order to reason about them logically.

These variables and their possible values are listed and described in Table 4.1; the variables that are directly observable are listed under the horizontal line.

RV	**Description**	**Values**
PID	Polled node identity; identity of the polled node	1; ... ; N
DR_i	Data rate; data rate of node i	Low; Medium; High
CQ_i	Current queue length; current queue length of node i	Small; Medium; Big
NP_i	Node polled; is node i getting polled	No; Yes
NQ_i	New queue length; queue length of node i after the next node has been polled	Small; Medium; Big
$NTSP_i$	New time elapsed since polled; time elapsed since node i was polled, after the next node has been polled	Short; Medium; Long
pqs_i	Previous queue length sent; number of packets (length of queue) that node i transmitted when it was previously polled	Small; Medium; Big
pwt_i	Previous waiting time; time duration node i had to wait before it was previously polled	Short; Medium; Long
tsp_i	Time elapsed since polled; time elapsed since node i has been polled	Short; Medium; Long

Table 4.1: RVs used in the decision cluster graph for determining which node the master node should poll. Observed variables are listed under the horizontal line.

The cut-off values used to discretize the observed continuous variables were determined heuristically; the cut-off values are:

$$Queue_{max} = \text{Maximum queue length a node is allowed to transmit per poll}$$

$$pqs_i = \begin{cases} \text{Small} & \text{if} \quad pqs_i < 0.3 \cdot Queue_{max} \\ \text{Medium} & \text{if} \quad 0.3 \cdot Queue_{max} < pqs_i < 0.6 \cdot Queue_{max} \\ \text{Big} & \text{if} \quad 0.6 \cdot Queue_{max} < pqs_i \end{cases} \tag{4.1}$$

$$Time_{max} = \text{Time required to poll and receive data from all slave nodes}$$

$$pwt_i = \begin{cases} \text{Short} & \text{if} \quad pwt_i < 0.4 \cdot Time_{total} \\ \text{Medium} & \text{if} \quad 0.4 \cdot Time_{total} < pwt_i < 0.9 \cdot Time_{total} \\ \text{Long} & \text{if} \quad 0.9 \cdot Time_{total} < pwt_i \end{cases} \tag{4.2}$$

$$tsp_i = \begin{cases} \text{Short} & \text{if} \quad tsp_i < 0.4 \cdot Time_{total} \\ \text{Medium} & \text{if} \quad 0.4 \cdot Time_{total} < tsp_i < 0.9 \cdot Time_{total} \\ \text{Long} & \text{if} \quad 0.9 \cdot Time_{total} < tsp_i \end{cases} \tag{4.3}$$

4.2.2 Decision Potentials

A decision cluster graph requires probability and utility factors in order to estimate probabilities and make decisions. Probability factors are advantageous since it grants the ability to add logical functions and dependencies between variables. Utility factors enable us to define how preferable the outcomes of RVs are. A decision potential, which is used in a decisions cluster graph, is a combination of a probability and utility factor (as described in Section 3.1.4).

The following logic was used to determine the decision potentials to be used in the decision cluster graph:

1. We want to ensure that all slave nodes are in the most preferred condition after the master node polls the next node. Thus we create a utility factor for each node, describing how preferable the state of that node is (after polling has occurred). The state of a node i is dependent on NQ_i (an indication that the node is becoming congested) and $NTSP_i$ (it influences the latency of packets and it is an indication that the node is being treated fairly). The utility factor for node i should have the following properties:

    - The longer $NTSP_i$, the worse the state of node i; the node has to wait longer to transmit its data (if it has data packets in its queue), increasing latency. It also indicate that the node is being treated unfairly.
    - The bigger NQ_i, the worse the state of node i since the node can become congested.
    - As long as NQ_i is small, a slightly longer $NTSP_i$ is still acceptable (if there are no data packets in the queue, it does not make a difference if

the node is not polled). However, $NTSP_i$ should never become too long (the node is being treated unfairly).

These properties lead to the heuristic utility factor (converted to a decision potential) listed in Table 4.2. All the RVs are discrete, thus it can happen (with a reasonable probability) that the state of different nodes can be exactly the same. Therefore we add a small amount of random noise to the utility values of the different nodes to distinguish between equal nodes.

$NTSP_i$	NQ_i	$\rho(NTSP_i\ ,\ NQ_i)$	$\mu(NTSP_i\ ,\ NQ_i)$
Short	Small	1.0	100
Short	Medium	1.0	-5
Short	Big	1.0	-40
Medium	Small	1.0	10
Medium	Medium	1.0	-10
Medium	Big	1.0	-60
Long	Small	1.0	-80
Long	Medium	1.0	-90
Long	Big	1.0	-100

Table 4.2: Decision Potential $\gamma(NTSP_i\ ,\ NQ_i)$, defining the utilities for the different values of $NTSP_i$ and NQ_i; describing how preferable the outcomes of the RVs are.

2. A probability factor is used to describe how polling node i influences its new time elapsed since it received a poll. If node i was polled, $NTSP_i$ is short; if not, $NTSP_i$ is most probably the same as tsp_i or slightly longer.

$$NTSP_i = \begin{cases} \text{Short} & \text{if} \quad NP_i = \text{Yes} \\ tsp_i & \text{if} \quad NP_i = \text{No} \end{cases} \tag{4.4}$$

This logic results in the heuristic probability factor (converted to a decision potential) listed in Table 4.3.

3. A probability factor is used to describe how polling node i influences its queue length. If node i was polled, NQ_i is small; if not, NQ_i is most probably the same as CQ_i or slightly bigger.

$$NQ_i = \begin{cases} \text{Small} & \text{if} \quad NP_i = \text{Yes} \\ CQ_i & \text{if} \quad NP_i = \text{No} \end{cases} \tag{4.5}$$

This logic results in the heuristic probability factor (converted to a decision potential) listed in Table 4.4.

NP_i	tsp_i	$NTSP_i$	$\rho(NTSP_i \mid NP_i\ ,\ tsp_i)$
No	Short	Short	0.85
No	Short	Medium	0.14
No	Short	Long	0.01
No	Medium	Short	0.0
No	Medium	Medium	0.85
No	Medium	Long	0.15
No	Long	Short	0.0
No	Long	Medium	0.0
No	Long	Long	1.0
Yes	Short	Short	1.0
Yes	Short	Medium	0.0
Yes	Short	Long	0.0
Yes	Medium	Short	1.0
Yes	Medium	Medium	0.0
Yes	Medium	Long	0.0
Yes	Long	Short	1.0
Yes	Long	Medium	0.0
Yes	Long	Long	0.0

Table 4.3: Decision Potential $\gamma(NTSP_i \mid NP_i\ ,\ tsp_i)$, using probabilities to describe how polling node i influences $NTSP_i$. Note that polling node i result in a short $NTSP_i$, while if node i is not polled, $NTSP_i$ is most probably the same as tsp_i, or slightly longer; as in equation (4.4).

NP_i	CQ_i	NQ_i	$\rho(NQ_i \mid NP_i\ ,\ CQ_i)$
No	Small	Small	0.85
No	Small	Medium	0.14
No	Small	Big	0.01
No	Medium	Small	0.0
No	Medium	Medium	0.85
No	Medium	Big	0.15
No	Big	Small	0.0
No	Big	Medium	0.0
No	Big	Big	1.0
Yes	Small	Small	0.9
Yes	Small	Medium	0.09
Yes	Small	Big	0.01
Yes	Medium	Small	0.9
Yes	Medium	Medium	0.09
Yes	Medium	Big	0.01
Yes	Big	Small	0.9
Yes	Big	Medium	0.09
Yes	Big	Big	0.01

Table 4.4: Decision Potential $\gamma(NQ_i|NP_i\ ,\ CQ_i)$, using probabilities to describe how polling node i influences $NTSP_i$. Note that polling node i result in a small NQ_i, while if node i is not polled, $NTSP_i$ is most probably the same as CQ_i, or slightly bigger; as in equation (4.5).

4. A probability factor is used to reason logically about CQ_i.
Since number of packets generated = (packets/second) · (time elapsed), DR_i and tsp_i should influence CQ_i as follows:

   - The higher DR_i, the higher the probability of a bigger CQ_i.
   - The longer longer tsp_i, the higher the probability of a bigger CQ_i.

$$CQ_i = \begin{cases} \text{Small} & \text{if} \quad DR_i = \text{Low}; \quad tsp_i = \text{Short} \\ \pm\,\text{Medium} & \text{if} \quad DR_i = \text{Low}; \quad tsp_i = \text{Medium} \\ \text{Big} & \text{if} \quad DR_i = \text{Low}; \quad tsp_i = \text{Long} \\ \pm\,\text{Small} & \text{if} \quad DR_i = \text{Medium}; \quad tsp_i = \text{Short} \\ \pm\,\text{Medium} & \text{if} \quad DR_i = \text{Medium}; \quad tsp_i = \text{Medium} \\ \text{Big} & \text{if} \quad DR_i = \text{Medium}; \quad tsp_i = \text{Long} \\ \text{Small} - \text{Medium} & \text{if} \quad DR_i = \text{High}; \quad tsp_i = \text{Short} \\ \text{Medium} - \text{Big} & \text{if} \quad DR_i = \text{High}; \quad tsp_i = \text{Medium} \\ \text{Big} & \text{if} \quad DR_i = \text{High}; \quad tsp_i = \text{Long} \end{cases} \tag{4.6}$$

This logic results in the heuristic probability factor (converted to a decision potential) listed in Table 4.5.

5. A probability factor is used to reason logically about DR_i. Since data rate = $\frac{\text{number of packets generated}}{\text{time elapsed}}$, pqs_i and pwt_i should influence DR_i as follows:

   - The bigger pqs_i, the higher the probability of a higher DR_i.
   - The longer pwt_i, the higher the probability of a lower DR_i.

$$DR_i = \begin{cases} ? & \text{if} \quad pqs_i = \text{Small}; \quad pwt_i = \text{Short} \\ \pm\,\text{Low} & \text{if} \quad pqs_i = \text{Small}; \quad pwt_i = \text{Medium} \\ \text{Low} & \text{if} \quad pqs_i = \text{Small}; \quad pwt_i = \text{Long} \\ \pm\,\text{High} & \text{if} \quad pqs_i = \text{Medium}; \quad pwt_i = \text{Short} \\ ? & \text{if} \quad pqs_i = \text{Medium}; \quad pwt_i = \text{Medium} \\ \pm\,\text{Medium} & \text{if} \quad pqs_i = \text{Medium}; \quad pwt_i = \text{Long} \\ \text{High} & \text{if} \quad pqs_i = \text{Big}; \quad pwt_i = \text{Short} \\ \pm\,\text{High} & \text{if} \quad pqs_i = \text{Big}; \quad pwt_i = \text{Medium} \\ \pm\,\text{High} & \text{if} \quad pqs_i = \text{Big}; \quad pwt_i = \text{Long} \end{cases} \tag{4.7}$$

This logic results in the heuristic probability factor (converted to a decision potential) listed in Table 4.6.

DR_i	tsp_i	CQ_i	$\rho(CQ_i \| DR_i\ ,\ tsp_i)$
Low	Short	Small	0.9
Low	Short	Medium	0.09
Low	Short	Big	0.01
Low	Medium	Small	0.25
Low	Medium	Medium	0.6
Low	Medium	Big	0.15
Low	Long	Small	0.05
Low	Long	Medium	0.1
Low	Long	Big	0.85
Medium	Short	Small	0.6
Medium	Short	Medium	0.3
Medium	Short	Big	0.1
Medium	Medium	Small	0.1
Medium	Medium	Medium	0.65
Medium	Medium	Big	0.25
Medium	Long	Small	0.01
Medium	Long	Medium	0.09
Medium	Long	Big	0.9
High	Short	Small	0.4
High	Short	Medium	0.4
High	Short	Big	0.2
High	Medium	Small	0.05
High	Medium	Medium	0.4
High	Medium	Big	0.55
High	Long	Small	0.01
High	Long	Medium	0.04
High	Long	Big	0.95

Table 4.5: Decision Potential $\gamma(CQ_i|DR_i\ ,\ tsp_i)$, using probabilities to describe how DR_i and tsp_i influences CQ_i. Further details can be found in the main text and equation (4.6).

pqs_i	pwt_i	DR_i	$\rho(DR_i \mid pqs_i\ ,\ pwt_i)$
Small	Short	Low	0.3
Small	Short	Medium	0.4
Small	Short	High	0.3
Small	Medium	Low	0.6
Small	Medium	Medium	0.3
Small	Medium	High	0.1
Small	Long	Low	0.9
Small	Long	Medium	0.09
Small	Long	High	0.01
Medium	Short	Low	0.1
Medium	Short	Medium	0.2
Medium	Short	High	0.7
Medium	Medium	Low	0.3
Medium	Medium	Medium	0.4
Medium	Medium	High	0.3
Medium	Long	Low	0.3
Medium	Long	Medium	0.6
Medium	Long	High	0.1
Big	Short	Low	0.01
Big	Short	Medium	0.09
Big	Short	High	0.9
Big	Medium	Low	0.08
Big	Medium	Medium	0.2
Big	Medium	High	0.72
Big	Long	Low	0.15
Big	Long	Medium	0.25
Big	Long	High	0.6

Table 4.6: Decision Potential $\gamma(DR_i \mid pqs_i\ ,\ pwt_i)$, using probabilities to describe how pqs_i and pwt_i influences DR_i. Further details can be found in the main text and equation (4.7).

6. The decision variable PID state which node to poll. The initial decision strategy is indecisive; all slave nodes are equally likely to be polled (Table 4.7).

PID	$\rho(PID)$
0	1.0
$\vdots$	$\vdots$
i	1.0
$\vdots$	$\vdots$
N	1.0

Table 4.7: Decision Potential $\gamma(PID)$, the default decision strategy for PID; all slave nodes are equally likely to be polled.

7. A probability factor is used to determine the value of NP_i from PID, splitting PID into N variables. The RVs associated with node i are dependent on whether node i is being polled; if it is not polled, the identity of the node (which is polled) is irrelevant. By doing this, it is possible to reduce the size of other factors (NP_i has 2 possible values, while PID has N). Thus the probability factor enforces the following logic:

$$NP_i = \begin{cases} \text{No} & \text{if} \quad PID \neq i \\ \text{Yes} & \text{if} \quad PID = i \end{cases} \tag{4.8}$$

This creates the probability factor (converted decision potential) found in Table 4.8.

PID	NP_i	$\rho(NP_i\|PID)$
0	No	1.0
0	Yes	0.0
$\vdots$	$\vdots$	$\vdots$
i	No	0.0
i	Yes	1.0
$\vdots$	$\vdots$	$\vdots$
N	No	1.0
N	Yes	0.0

Table 4.8: Decision Potential $\gamma(NP_i|PID)$, using probabilities to split PID into N variables. Thus NP_i states that node i is not polled if $PID \neq i$, else node i is polled, as in equation 4.8.

4.2.3 Polling with a Decision Cluster Graph

In order to decide which node should be polled, Algorithm 1 is used to assign a decision strategy for PID. Executing Algorithm 1:

1. RVs and decision potential are defined in Section 4.2.1 and 4.2.2.
2. A LIMID (Fig. 4.1) is created from the statistical dependencies between the RVs found in the decision potentials of Section 4.2.2.
3. Since there is only one decision to be made, the relevance graph is extremely simple; the (acyclic) relevance graph contains a single node.
4. The decision potentials of Section 4.2.2 are used to create the decision cluster graph (Fig. 4.2).
5. Due to the simplistic relevance graph, the remaining part of Algorithm 1 is:
    a) Performing belief-propagation
    b) Assigning a decision strategy to PID; dictating which node to poll

Note that the decision cluster graph contains loops, which increases the duration of the computations. Nonetheless, the algorithm converges to a solution quite quickly.

In order to compare the performance of the decision cluster graph to current theory, we implemented a strong JT from the same RVs and decision potentials. The resulting strong JT is illustrated in Fig. 4.3. Deciding which node to poll is similar to the decision cluster graph, propagating messages from the leaves of the tree towards the root of the tree (cluster PID) and assigning a decision strategy to PID.

The extra computations that accompany decisions, utilities and loopy cluster graphs, as well as the flaws of decision cluster graphs, led to creating a solution (Section 4.3) for this problem using a simplistic cluster graph (without decisions and utilities); deciding which node to poll based on probabilities.

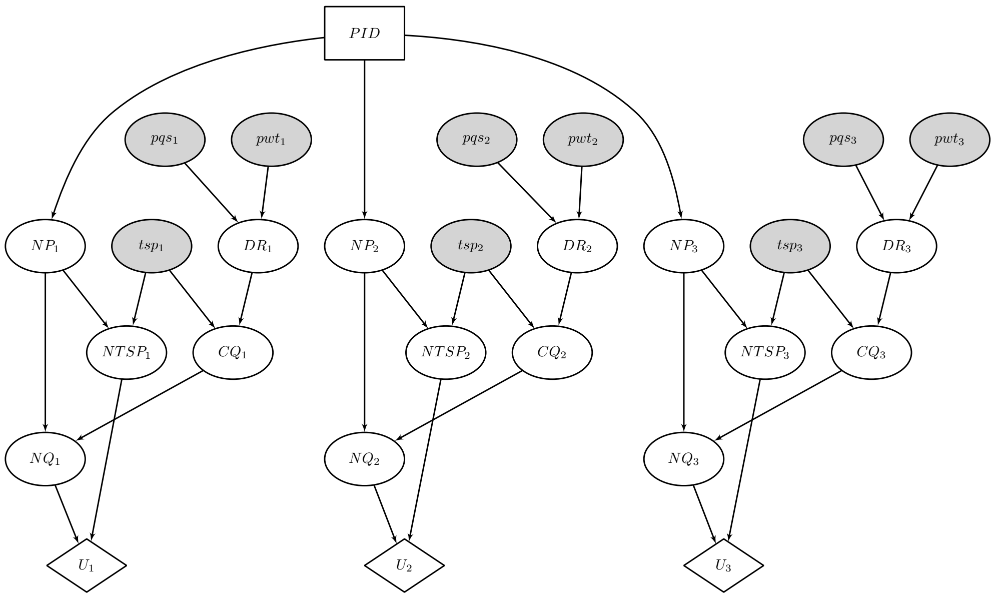


Figure 4.1: LIMID for the decision cluster graph augmented Round Robin MAC protocol, illustrating the statistical dependencies between RVs for a network with 3 slave nodes. Observed RVs are shaded.

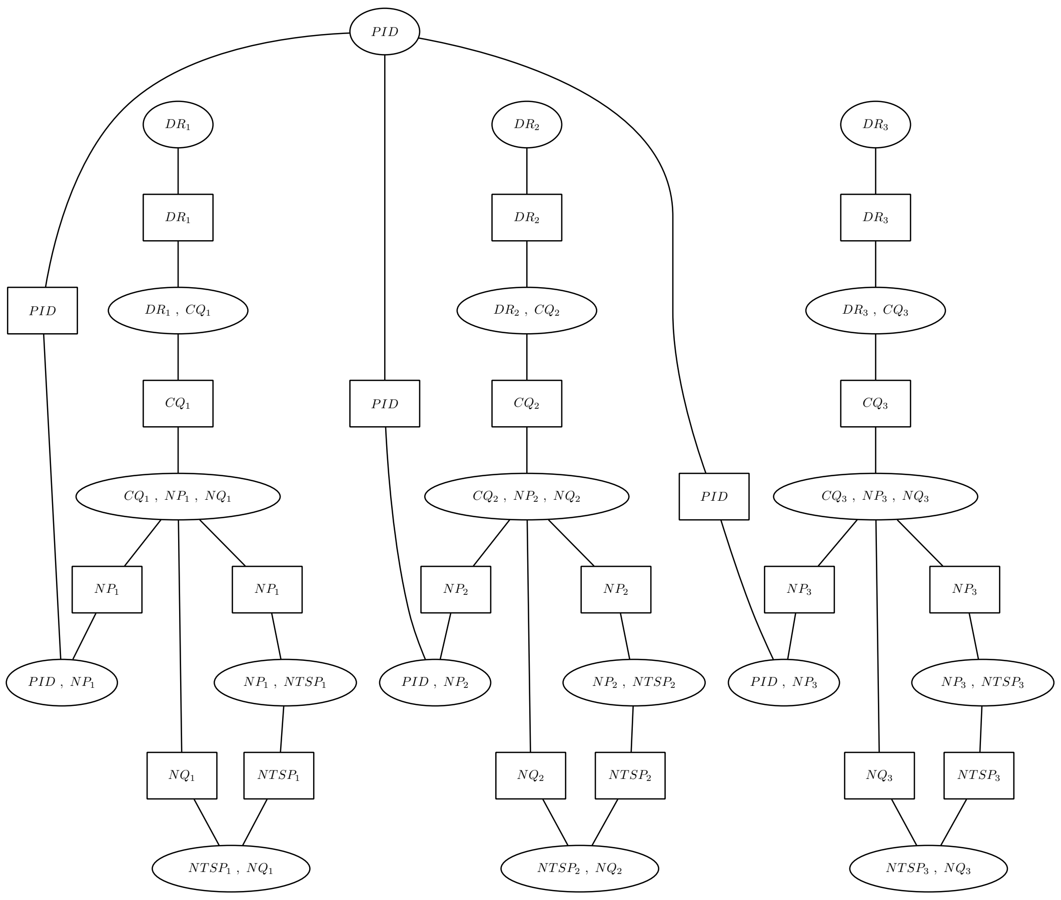


Figure 4.2: Decision cluster graph for the decision cluster graph augmented Round Robin MAC protocol; used to determine which slave node to poll (for a network with 3 slave nodes). Note that the decision cluster graph contains loops, resulting in slightly longer and inaccurate calculations.

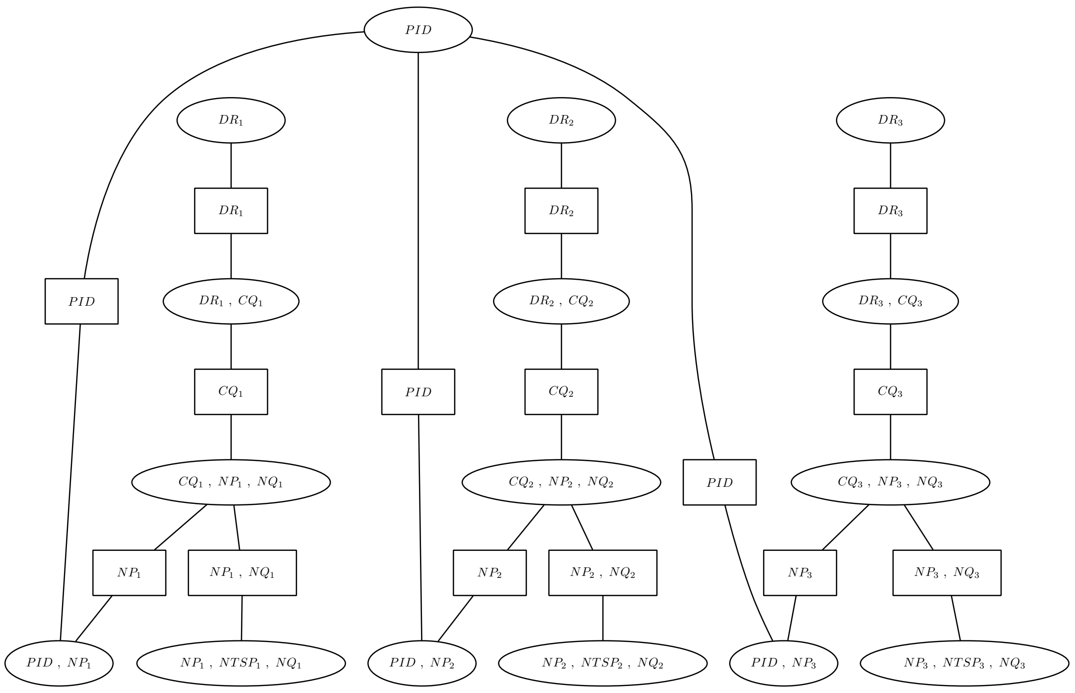


Figure 4.3: Strong JT for the strong JT augmented Round Robin MAC protocol; used to determine which slave node to poll (for a network with 3 slave nodes). Note that since this is a JT, the graph does not contain loops, resulting in quick and accurate calculations.

4.3 Round Robin Protocol Augmented with a Cluster Graph

The objective of this method is to have a simpler and computationally quicker approach than the decision cluster graph approach. Since we are solving the same problem of deciding which slave node to poll, there are many similarities to the decision cluster graph approach (Section 4.2).

This approach is also a practical example for another method to make decisions under uncertainty: making a decision based on the probabilities associated with certain RV(s). The advantage of this method is that it has simpler calculations and it avoids some of the flaws associated with decision cluster graphs.

In order to decide which slave node to poll (according to estimated probabilities), it is once again necessary to first identify the relevant variables.

4.3.1 Relevant Variables

There are many similarities between this method and the decision cluster graph approach, particularly in respect of RVs and their dependencies. However, the main difference is that for this method we introduce another RV for each slave node, stating the overall condition of the node (NC_i). These RVs are used to determine which node to poll, polling the node that has the highest probability to be in a poor condition. These RVs can be seen as the replacement for the utility factors.

The following variables from the decision cluster graph approach are also relevant for this approach:

In order to improve the performance of the protocol, we want to avoid polling nodes that do not have data to transmit; thus we estimate the queue length for each node (CQ_i). Also, a node with a long queue becomes congested, resulting in increased latencies for the packets.

We also want to ensure fairness in the network, thus we log the time elapsed since a node was polled (tsp_i) (if a node was polled long ago, it could indicate that the node is being treated unfairly). The time elapsed since a node was polled also influences performance, since it influences the latency of the packets (but only if there are packets in the node's queue).

The queue size of a node is dependent on the time elapsed since it was polled, but also on the data rate of the node (DR_i). We can estimate a node's data rate from the following observable variables: the previous number of packets that the node transmitted (pqs_i), and how long the node had to wait to be polled in order to transmit said packet(s) (pwt_i).

Once again, we used discrete variables, since the calculations are much simpler and it is easier to create logical functions between the variables. Although the observed variables are actually continuous, we discretize them in 3 levels in order to reason about them logically.

These variables and their possible values are listed and described in Table 4.9; the variables that are directly observable are listed under the horizontal line.

RV	**Description**	**Values**
NC_i	Node condition; what is the overall condition of node i?	Poor; Excellent
CQ_i	Current queue length; current queue length of node i	Small; Medium; Big
DR_i	Data rate; data rate of node i	Low; Medium; High
pqs_i	Previous queue length sent; number of packets (length of queue) that node i transmitted when it was previously polled	Small; Medium; Big
pwt_i	Previous waiting time; time duration node i had to wait before it was previously polled	Short; Medium; Long
tsp_i	Time elapsed since polled; time elapsed since node i has been polled	Short; Medium; Long

Table 4.9: Relevant RVs for deciding which node to poll in the cluster graph augmented Round Robin MAC protocol. Observed variables are listed under the horizontal line.

The same heuristic cut-off values used to discretize the observed continuous variables were used. The cut-off values are:

$$Queue_{max} = \text{Maximum queue length a node is allowed to transmit per poll}$$

$$pqs_i = \begin{cases} \text{Small} & \text{if} \quad pqs_i < 0.3 \cdot Queue_{max} \\ \text{Medium} & \text{if} \quad 0.3 \cdot Queue_{max} < pqs_i < 0.6 \cdot Queue_{max} \\ \text{Big} & \text{if} \quad 0.6 \cdot Queue_{max} < pqs_i \end{cases} \tag{4.9}$$

$$Time_{max} = \text{Time required to poll and receive data from all slave nodes}$$

$$pwt_i = \begin{cases} \text{Short} & \text{if} \quad pwt_i < 0.4 \cdot Time_{total} \\ \text{Medium} & \text{if} \quad 0.4 \cdot Time_{total} < pwt_i < 0.9 \cdot Time_{total} \\ \text{Long} & \text{if} \quad 0.9 \cdot Time_{total} < pwt_i \end{cases} \tag{4.10}$$

$$tsp_i = \begin{cases} \text{Short} & \text{if} \quad tsp_i < 0.4 \cdot Time_{total} \\ \text{Medium} & \text{if} \quad 0.4 \cdot Time_{total} < tsp_i < 0.9 \cdot Time_{total} \\ \text{Long} & \text{if} \quad 0.9 \cdot Time_{total} < tsp_i \end{cases} \tag{4.11}$$

4.3.2 Probability Factors

A cluster graph requires probability factors in order to estimate probabilities. Probability factors are advantageous, since they grant the ability to add logical functions and dependencies between variables.

The following logic was used to determine the probability factors to be used in the cluster graph (once again, most of the logic is the same as the decision cluster graph approach):

1. The same logic as in Table 4.5 is used to reason logically about CQ_i: Since number of packets generated = (packets/second) · (time elapsed), DR_i and tsp_i should influence CQ_i as follows:

   - The higher DR_i, the higher the probability of a bigger CQ_i.
   - The longer longer tsp_i, the higher the probability of a bigger CQ_i.

$$CQ_i = \begin{cases} \text{Small} & \text{if} \quad DR_i = \text{Low}; \quad tsp_i = \text{Short} \\ \pm\,\text{Medium} & \text{if} \quad DR_i = \text{Low}; \quad tsp_i = \text{Medium} \\ \text{Big} & \text{if} \quad DR_i = \text{Low}; \quad tsp_i = \text{Long} \\ \pm\,\text{Small} & \text{if} \quad DR_i = \text{Medium}; \quad tsp_i = \text{Short} \\ \pm\,\text{Medium} & \text{if} \quad DR_i = \text{Medium}; \quad tsp_i = \text{Medium} \\ \text{Big} & \text{if} \quad DR_i = \text{Medium}; \quad tsp_i = \text{Long} \\ \text{Small} - \text{Medium} & \text{if} \quad DR_i = \text{High}; \quad tsp_i = \text{Short} \\ \text{Medium} - \text{Big} & \text{if} \quad DR_i = \text{High}; \quad tsp_i = \text{Medium} \\ \text{Big} & \text{if} \quad DR_i = \text{High}; \quad tsp_i = \text{Long} \end{cases} \tag{4.12}$$

This logic results in the heuristic probability factor listed in Table 4.10.

DR_i	tsp_i	CQ_i	$\rho(CQ_i \vert DR_i\ ,\ tsp_i)$
Low	Short	Small	0.9
Low	Short	Medium	0.09
Low	Short	Big	0.01
Low	Medium	Small	0.25
Low	Medium	Medium	0.6
Low	Medium	Big	0.15
Low	Long	Small	0.05
Low	Long	Medium	0.1
Low	Long	Big	0.85
Medium	Short	Small	0.6
Medium	Short	Medium	0.3
Medium	Short	Big	0.1
Medium	Medium	Small	0.1
Medium	Medium	Medium	0.65
Medium	Medium	Big	0.25
Medium	Long	Small	0.01
Medium	Long	Medium	0.09
Medium	Long	Big	0.9
High	Short	Small	0.4
High	Short	Medium	0.4
High	Short	Big	0.2
High	Medium	Small	0.05
High	Medium	Medium	0.4
High	Medium	Big	0.55
High	Long	Small	0.01
High	Long	Medium	0.04
High	Long	Big	0.95

Table 4.10: Probability factor $\rho(CQ_i|DR_i\ ,\ tsp_i)$, using probabilities to describe how DR_i and tsp_i influence CQ_i. Note that the same logic as in Table 4.5 is used; further details can be found in the main text and equation (4.12).

2. The same logic as in Table 4.6 is used to reason logically about DR_i: Since data rate $= \frac{\text{number of packets generated}}{\text{time elapsed}}$, pqs_i and pwt_i should influence DR_i as follows:

   - The bigger pqs_i, the higher the probability of a higher DR_i.
   - The longer pwt_i, the higher the probability of a lower DR_i.

$$DR_i = \begin{cases} ? & \text{if} \quad pqs_i = \text{Small}; \quad pwt_i = \text{Short} \\ \pm\,\text{Low} & \text{if} \quad pqs_i = \text{Small}; \quad pwt_i = \text{Medium} \\ \text{Low} & \text{if} \quad pqs_i = \text{Small}; \quad pwt_i = \text{Long} \\ \pm\,\text{High} & \text{if} \quad pqs_i = \text{Medium}; \quad pwt_i = \text{Short} \\ ? & \text{if} \quad pqs_i = \text{Medium}; \quad pwt_i = \text{Medium} \\ \pm\,\text{Medium} & \text{if} \quad pqs_i = \text{Medium}; \quad pwt_i = \text{Long} \\ \text{High} & \text{if} \quad pqs_i = \text{Big}; \quad pwt_i = \text{Short} \\ \pm\,\text{High} & \text{if} \quad pqs_i = \text{Big}; \quad pwt_i = \text{Medium} \\ \pm\,\text{High} & \text{if} \quad pqs_i = \text{Big}; \quad pwt_i = \text{Long} \end{cases} \tag{4.13}$$

   This logic results in the heuristic probability factor listed in Table 4.11.

3. A probability factor is used to reason logically about NC_i. Since NC_i describes the condition of node i, NC_i is dependent on CQ_i (an indication that the node is congested) and tsp_i (it influences the latency of packets and is an indication that the node is being treated fairly). The probability factor should have the following characteristics:

   - The longer tsp_i, the worse the condition of node i; the node has waited longer to transmit its data (if it has packets in its queue), increasing latency. It also indicates that the node is being treated unfairly.
   - The bigger CQ_i, the worse the condition of node i since the node could be congested.
   - As long as CQ_i is small, a slightly longer tsp_i is still acceptable. However, tsp_i should never be too long.

$$NC_i = \begin{cases} \text{Excellent} & \text{if} \quad tsp_i = \text{Short}; \quad CQ_i = \text{Small} \\ \pm\,\text{Poor} & \text{if} \quad tsp_i = \text{Short}; \quad CQ_i = \text{Medium} \\ \text{Poor} & \text{if} \quad tsp_i = \text{Short}; \quad CQ_i = \text{Big} \\ \pm\,\text{Excellent} & \text{if} \quad tsp_i = \text{Medium}; \quad CQ_i = \text{Small} \\ \pm\,\text{Poor} & \text{if} \quad tsp_i = \text{Medium}; \quad CQ_i = \text{Medium} \\ \text{Poor} & \text{if} \quad tsp_i = \text{Medium}; \quad CQ_i = \text{Big} \\ \text{Poor} & \text{if} \quad tsp_i = \text{Long}; \quad CQ_i = \text{Small} \\ \text{Poor} & \text{if} \quad tsp_i = \text{Long}; \quad CQ_i = \text{Medium} \\ \text{Poor} & \text{if} \quad tsp_i = \text{Long}; \quad CQ_i = \text{Big} \end{cases} \tag{4.14}$$

This logic results in the heuristic probability factor listed in Table 4.12. All the RVs are discrete, thus it can happen (with a reasonable probability) that the condition of different nodes can be exactly the same. Therefore, to distinguish between different nodes, we add a small amount of random noise to the probability values in Table 4.12 to distinguish between equal nodes.

pqs_i	pwt_i	DR_i	$\rho(DR_i \vert pqs_i\ ,\ pwt_i)$
Small	Short	Low	0.3
Small	Short	Medium	0.4
Small	Short	High	0.3
Small	Medium	Low	0.6
Small	Medium	Medium	0.3
Small	Medium	High	0.1
Small	Long	Low	0.9
Small	Long	Medium	0.09
Small	Long	High	0.01
Medium	Short	Low	0.1
Medium	Short	Medium	0.2
Medium	Short	High	0.7
Medium	Medium	Low	0.3
Medium	Medium	Medium	0.4
Medium	Medium	High	0.3
Medium	Long	Low	0.3
Medium	Long	Medium	0.6
Medium	Long	High	0.1
Big	Short	Low	0.01
Big	Short	Medium	0.09
Big	Short	High	0.9
Big	Medium	Low	0.08
Big	Medium	Medium	0.2
Big	Medium	High	0.72
Big	Long	Low	0.15
Big	Long	Medium	0.25
Big	Long	High	0.6

Table 4.11: Probability factor $\rho(DR_i|pqs_i\ ,\ pwt_i)$, using probabilities to describe how pqs_i and pwt_i influence DR_i. Note that the same logic as in Table 4.6 is used; further details can be found in the main text and equation (4.13).

tsp_i	CQ_i	NC_i	$\rho(NC_i \vert tsp_i\ ,\ CQ_i)$
Short	Small	Poor	0.001
Short	Small	Excellent	0.999
Short	Medium	Poor	0.6
Short	Medium	Excellent	0.4
Short	Big	Poor	0.85
Short	Big	Excellent	0.05
Medium	Small	Poor	0.3
Medium	Small	Excellent	0.7
Medium	Medium	Poor	0.7
Medium	Medium	Excellent	0.3
Medium	Big	Poor	0.9
Medium	Big	Excellent	0.1
Long	Small	Poor	0.9
Long	Small	Excellent	0.1
Long	Medium	Poor	0.95
Long	Medium	Excellent	0.05
Long	Big	Poor	0.999
Long	Big	Excellent	0.001

Table 4.12: Probability factor $\rho(NC_i|tsp_i\ ,\ CQ_i)$, using probabilities to describe how tsp_i and CQ_i influence NC_i. Further details can be found in the main text and equation (4.14).

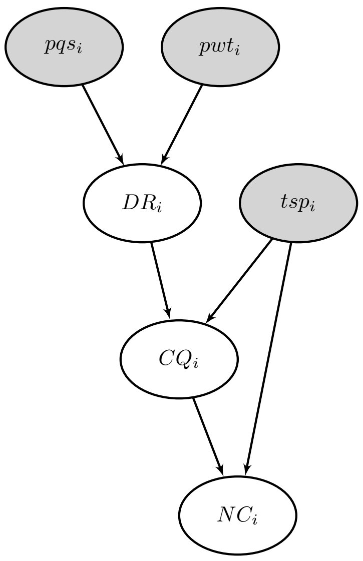
4.3.3 Polling with Probabilities

In order to improve the condition of the network, the master node has to poll the slave node that is probably in the worst condition. Therefore the master node requires the probabilities for all the NC_i variables. Cluster graphs are used to calculate these probabilities efficiently.

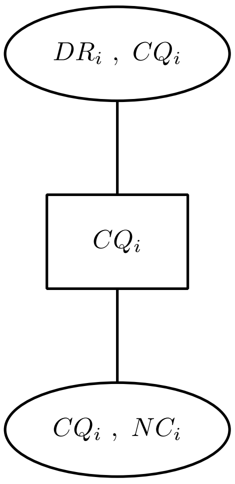
A Bayes network (Fig. 4.4a) is used to illustrate the statistical dependencies between the RVs found in the probability factors of Section 4.3.2. Since there are no statistical dependencies between the variables of different slave nodes, a Bayes network for each slave node is created.

A cluster graph (Fig. 4.4b) for each slave node i is created from the probability factors of Section 4.3.2, and belief-propagation is used to calculate the probabilities of NC_i. In order to improve the condition of the network, the master node polls the slave node with the highest probability of a poor NC_i.

Note that the cluster graph is a JT since it does not contain loops, resulting in quick and accurate calculations.



(a) Bayes network illustrating the statistical dependencies between the RVs of the cluster graph augmented Round Robin MAC protocol; shading the observed RVs.



(b) Cluster graph used to estimate the probability that node i is in a poor condition (for the cluster graph augmented Round Robin MAC protocol). Note that the cluster graph contains no loops, thus it is a junction tree, resulting in quick and accurate calculations.

Figure 4.4

4.4 Conclusions and Recommendations

The following observations and recommendations about these models can be made:

- Assigning a level (low, medium, high) to the observed variables destroys some information and degrades the performance slightly. This also increases the chance for nodes to be regarded as exactly equal. To minimize this data-loss, it is possible to create a cluster graph with discrete RVs only, while the observed RVs remain continuous since observed RVs are effectively removed from the cluster graph. This is achieved by altering the probabilities of the RVs that depend on the observed RVs according to the value of the observed RVs. We applied this method for augmenting the CSMA/CA MAC protocol (Chapter 5). We did not implement this technique for the Round Robin MAC methods, illustrating that it is not strictly necessary, and a simpler approach is feasible. However, it is recommended for future work that requires a more accurate and complex model.

- In order to improve the accuracy of the estimations, more complex models can be created. For example, a Markov chain can be used to model changes over time. Interactions between nodes can also be modelled (as transmissions are generally a 2-way communication). However, it is important to bear the issues of a loopy decision cluster graph in mind. A complex model can also degrade performance, as loopy cluster graphs do not guarantee convergence. Also, the advantage of being able to calculate probabilities quickly using a simpler model should not be underestimated.

- The decision potentials and probability factors of these methods were created heuristically and were by no means optimized. This project focussed on the feasibility of augmenting MAC protocols with PGMs, thus we did not optimize the methods for maximum performance. For future work we recommend optimizing these methods for real-world situations.

The simulations, results, and performance comparisons of these methods are covered in Chapter 6.

Chapter 5

Implementing a PGM in the CSMA/CA MAC Protocol

Augmenting a contention-based MAC protocol further illustrates the feasibility and advantages of augmenting MAC protocols with a PGM. We chose to augment the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC protocol, as it is used in the highly popular IEEE 802.11 standard. We augment the protocol by using a PGM to estimate the number of contending nodes and adapting the length of the contention window accordingly.

5.1 Protocol Background

The CSMA/CA MAC protocol is used in the IEEE 802.11 standard. This standard is widely adopted, and improving on it could therefore be beneficial to a wide range of applications. We consider specifically the Request-To-Send (RTS) Clear-To-Send (CTS) handshaking scheme. For a node to transmit data over the channel, it has to win channel access via contention. Contention is resolved by allowing participating nodes to randomly choose an idle time slot (between 0 and CW, the contention window) and transmitting an RTS message in that time slot. If a node chooses a unique slot, it wins channel access (the receiving node transmits a CTS message) and is allowed to transmit its data. If a node was unsuccessful, CW is doubled (up to CW_{max}) and the node has to contend again. CW is set to CW_{min} for each new packet. Clearly the value of CW determines the length and success of the contention period. These RTS and CTS messages contain information about the length of the upcoming transmission. Thus to avoid collisions, all non-participating nodes receiving either the RTS or CTS message defer their transmissions until that transmission completes. For in-depth background about the 802.11 DCF MAC protocol, see [6] and [7].

5.2 Objectives

The overall objective is to improve the performance of the CSMA/CA MAC protocol by augmenting the protocol with a PGM. The performance of the protocol can be improved by optimizing the contention period; a too-short contention period can result in many unsuccessful contention attempts, while a too-long contention period wastes time. Optimizing the contention period can be achieved by finding the optimal value for CW. However, the optimal value for CW is dependent on the number of nodes contending for channel access -- if there are only a few contending nodes, CW needs to be small to reduce time wastage, while if there are many contending nodes, CW needs to be large to improve the probability of success. Thus it is essential that each node estimates the number of contending nodes. From this estimation, the optimal value for CW can be determined.

5.3 Implementation

Optimizing the contention period is achieved by augmenting the CSMA/CA MAC protocol with a cluster graph, allowing each node (the calculating node) to estimate the number of contending nodes (Section 5.3.1) and adjusting its CW to the corresponding optimal value (Section 5.3.2).

Note that we make the assumption that all nodes in the network are within transmission range of one another. This assumption does not impact the protocol significantly, since in a typical wireless network most nodes are within transmission range of one another. Thus it is almost certain that a node is within transmission range of either the transmitting or receiving node. And since we use the collision-avoidance protocol (nodes that do not participate in a transmission but receive either the RTS- or CTS-message wait until the transmission is over), the effect of this assumption is negligible.

5.3.1 Estimating the number of contending nodes

The number of contending nodes can be estimated by first estimating the probability that each active neighbouring node (indexed i) wants to contend (a node is seen as active, in regard to transmitting, if it has shown activity in the last 200ms[1]). The total number of nodes contending for access to the channel is the sum of the individual nodes.

[1]This is a parameter choice (made heuristically) and can be changed. For our simulations, this allows for ≈ 200 successful transmissions.

For each active neighbouring node, a cluster graph is used to estimate the probability that said node wants to participate in the contention period. However, before we can create a cluster graph, it is necessary to identify the relevant random variables (RVs).

5.3.1.1 Relevant Variables

Our first objective is to estimate if node i (for all active neighbouring nodes) wants to participate in contending for access to the channel (CP_i).

Participation in the contention period for node i is dependent on the rate at which node i generates RTS-messages (RGR_i), and also if it is time to expect activity from node i (EX_i).

Expecting activity from node i is dependent on the time elapsed since node i made its previous transmission (tst_i).

The rate at which node i generates RTS messages is dependent on the data rate of node i (DR_i) and whether node i is successful at winning access to the channel (NS_i).

The data rate of node i is dependent on the number of RTS messages that the calculating node received from node i (nrs_i).

Whether node i is successful at winning access to the channel is dependent on whether contention in the network is happening successfully (SC_i) and whether the number of RTS and data messages that the calculating node received from node i match (MRD_i) (each data packet should have a matching RTS message).

Whether the number of RTS and data messages of node i match is dependent on the difference between the number of RTS and data messages that the calculating node received from node i (dif_i).

We use discrete variables for all unobserved RVs, since the calculations are much simpler and it is easier to create logical functions between the RVs. The observed variables remain continuous, since observing them effectively removes them from the cluster graph. We use functions to determine the probabilities for the RVs, which are dependent on the observed variables. The relevant variables and their possible values are listed and described in Table 5.1; the variables that are directly observable are listed under the horizontal line.

RV	Description	Values
CP_i	Contention participation; does node i want to participate in contending for access to the channel?	No; Yes
RGR_i	RTS message-generation rate of node i; the rate at which node i generates RTS messages	Low; High
EX_i	Is it time to expect activity from node i?	No; Yes
NS_i	Node Success; Is node i successful at winning access to the channel?	No; Yes
DR_i	Data rate of node i	Low; High
SC	Successful Contention; Is contention happening successfully in the network?	No; Yes
MRD_i	Match the number of RTS and data messages received from node i?	Match; Different
dif_i	Difference between the number of RTS and data messages received by the calculating node from node i	$[0, \infty)$
nrs_i	Number of RTS messages that the calculating node received from node i	$[0, \infty)$
tst_i	Time elapsed since previous transmission; time elapsed since node i made its previous (successful) transmission	$[0, \infty)$

Table 5.1: RVs used in the cluster graph augmented CSMA/CA MAC protocol to estimate the probability that node i wants to participate in contending for access to the channel. Observed variables are listed under the horizontal line.

In order to estimate the probabilities of some of the unobserved variables, some additional statistics about the calculating node are recorded. The following information is recorded periodically (over the active period for neighbouring nodes, in our case 200ms):

- Total number of RTS messages received from neighbouring nodes (number of successful attempts at contention for the neighbouring nodes).
- Number of CTS-message replies received from the node that is the destination of the data packet (number of successful attempts at contention for the calculating node).
- Number of observed collisions in the network (number of time slots in the contention period for which the channel was not idle but no discernible RTS message was received, plus the number of unsuccessful attempts at contention of the calculating node).

Since the continuous variables are observed, we use functions to determine the probabilities of the variables that depend on these observed variables. By doing this, the information loss associated with assigning a level to an observed variable is minimal. These are the heuristic equations that we use:

- Estimating $\rho(SC = \text{Yes})$ (probability that contention happens successfully in the network):

$$\begin{aligned}
&\text{number of successful attempts} = \\
&\qquad (\text{number of successful attempts of this node}) + \\
&\qquad (\text{number of successful attempts of neighbours}) \\
&\text{number of successful attempts} = \\
&\qquad (\text{number of CTS messages received}) + \\
&\qquad (\text{number of RTS messages received from neighbours}) \\
&\text{number of unsuccessful attempts} = 2 \times (\text{number of collisions}) \\
&\text{number of attempts} = \\
&\qquad (\text{number of successful attempts}) + \\
&\qquad (\text{number of unsuccessful attempts})
\end{aligned}$$

$$\rho(SC = \text{Yes}) = \frac{\text{number of successful attempts}}{\text{number of attempts}} \tag{5.1}$$

Note that equation (5.1) makes the assumption that there are exactly 2 nodes transmitting to create a collision.

- Estimating $\rho(MRD_i = \text{Different})$ (probability that the number of RTS and data messages received from node i match):

$$\rho(MRD_i = \text{Different}) = 1 - e^{-0.3*(dif_i)^2} \tag{5.2}$$

  Heuristic equation (5.2), illustrated in Fig. 5.1, should have the following properties:

  - $\rho(MRD_i = \text{Different}) = [0, 1)$ for $dif_i = [0, \infty)$
  - $\rho(MRD_i = \text{Different}) \approx 0.2$ for $dif_i = 1$ (low probability when dif_i is small)
  - $\rho(MRD_i = \text{Different}) \approx 0.9$ for $dif_i = 3$ (high probability when dif_i becomes big; a significant difference between the number of RTS and data messages received indicates that the number of RTS and data messages do not match)
  - Monotonically increasing function

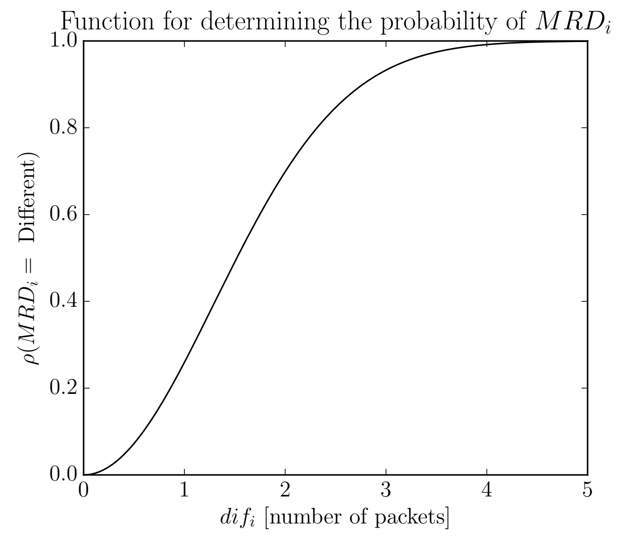


Figure 5.1: Heuristic function used to calculate $\rho(MRD_i = \text{Different})$ (equation 5.2). Note that $\rho(MRD_i = \text{Different}) = [0, 1)$ for $dif_i = [0, \infty)$; $\rho(MRD_i = \text{Different}) \approx 0.2$ for $dif_i = 1$; $\rho(MRD_i = \text{Different}) \approx 0.9$ for $dif_i = 3$.

- Estimating $\rho(EX_i = \text{Yes})$ (probability for expecting activity from node i) and $\rho(DR_i = \text{High})$ (probability that node i has a high data rate):

$$\begin{aligned}
&\text{Time required for a fair number of transmissions} = \\
&\qquad \text{Time required per transmission} \times \\
&\qquad \text{number of active neighbouring nodes} \\
&tst_{norm} = \frac{tst_i}{\text{Time required for a fair number of transmissions}}
\end{aligned}$$

$$\rho(EX_i = \text{Yes}) = 1 - e^{-2.0*(tst_{norm})} \tag{5.3}$$

$$\begin{aligned}
&\text{fair number of transmissions} = \\
&\qquad \text{active period duration} \div \\
&\qquad \text{Time required for a fair number of transmissions} \\
&nrs_{norm} = \frac{nrs_i}{\text{fair number of transmissions}}
\end{aligned}$$

$$\rho(DR_i = \text{High}) = 1 - e^{-2.0*(nrs_{norm})^2} \tag{5.4}$$

Heuristic equation (5.3), illustrated in Fig. 5.2, should have the following properties:

- $\rho(EX_i = \text{Yes}) = [0, 1)$ for $tst_{norm} = [0, \infty)$
- $\rho(EX_i = \text{Yes}) \approx 0.85$ for $tst_{norm} = 1$ (high probability when previous transmission becomes suspiciously long ago)
- Monotonically increasing function

Heuristic equation (5.4), illustrated in Fig. 5.3, should have the following properties:

- $\rho(DR_i = \text{High}) = [0, 1)$ for $nrs_{norm} = [0, \infty)$
- $\rho(DR_i = \text{High}) \approx 0.1$ for $nrs_{norm} = 0.25$ (low probability when only a few RTS messages are received)
- $\rho(DR_i = \text{High}) \approx 0.85$ for $nrs_{norm} = 1$ (high probability when the number of RTS messages received becomes significant)
- Monotonically increasing function

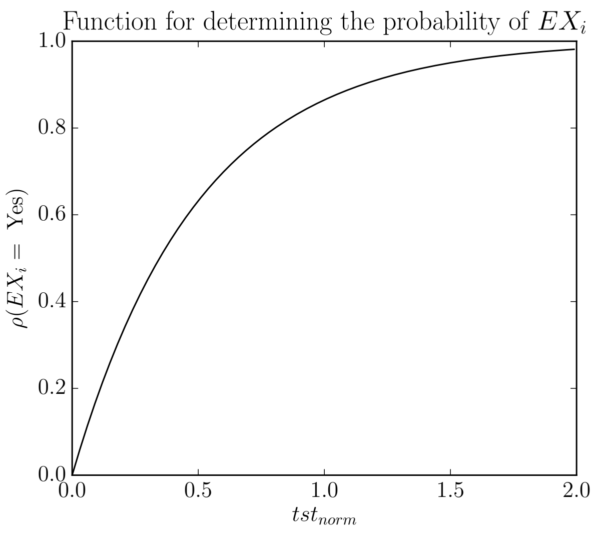


Figure 5.2: Heuristic function used to calculate $\rho(EX_i = \text{Yes})$ (equation 5.3). Note that $\rho(EX_i = \text{Yes}) = [0, 1)$ for $tst_{norm} = [0, \infty)$; $\rho(EX_i = \text{Yes}) \approx 0.85$ for $tst_{norm} = 1$.

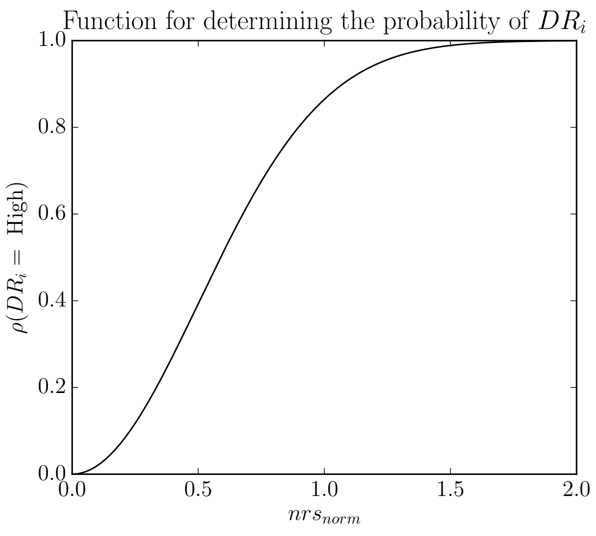


Figure 5.3: Function used to calculate $\rho(DR_i = \text{High})$ (equation 5.4). Note that $\rho(DR_i = \text{High}) = [0, 1)$ for $nrs_{norm} = [0, \infty)$; $\rho(DR_i = \text{High}) \approx 0.1$ for $nrs_{norm} = 0.25$; $\rho(DR_i = \text{High}) \approx 0.85$ for $nrs_{norm} = 1$.

These equations are not optimized, but small variations in the parameters have little effect on our method.

5.3.1.2 Probability Factors

In order to estimate if a neighbouring node wants to participate in contending for access to the channel, a cluster graph is used. However, a cluster graph requires probability factors in order to estimate probabilities. Probability factors are advantageous as they grant us the ability to add logical functions and dependencies between variables.

The following logic was used to determine the probability factors to be used in the cluster graph:

1. A node is not successful at winning access to the channel if the number of RTS and data messages received from the node do not match. However, if the number of RTS and data messages received from the node do match, the node is as successful as the current success rate of the network.

$$NS_i = \begin{cases} SC & \text{if } MRD_i = \text{Match} \\ \text{No} & \text{if } MRD_i = \text{Different} \end{cases} \tag{5.5}$$

   This logic result in the probability factor listed in Table 5.2.

MRD_i	SC	NS_i	$\rho(NS_i \vert MRD_i, SC)$
Match	No	No	1.0
Match	No	Yes	0.0
Match	Yes	No	0.0
Match	Yes	Yes	1.0
Different	No	No	1.0
Different	No	Yes	0.0
Different	Yes	No	1.0
Different	Yes	Yes	0.0

Table 5.2: Probability factor $\rho(NS_i|MRD_i, SC)$, using probabilities to describe how MRD_i and SC influence NS_i. Note that the logic of equation (5.5) is implemented, $NS_i = SC$ if MRD_i = Match, else $NS_i = No$ if MRD_i = Different.

2. The rate at which node i generates RTS messages is high if it is unsuccessful at winning access to the channel. If node i is successful, its RTS message-generation rate will correspond with its data rate.

$$RGR_i = \begin{cases} \text{High} & \text{if } NS_i = \text{No} \\ DR_i & \text{if } NS_i = \text{Yes} \end{cases} \tag{5.6}$$

   This logic results in the probability factor found in Table 5.3.

NS_i	DR_i	RGR_i	$\rho(RGR_i \mid NS_i, DR_i)$
No	Low	Low	0.0
No	Low	High	1.0
No	High	Low	0.0
No	High	High	1.0
Yes	Low	Low	1.0
Yes	Low	High	0.0
Yes	High	Low	0.0
Yes	High	High	1.0

Table 5.3: Probability factor $\rho(RGR_i|NS_i, DR_i)$, using probabilities to describe how NS_i and DR_i influence RGR_i. Note that the logic of equation (5.6) is implemented, $RGR_i = \text{High}$ if $NS_i = \text{No}$, else $RGR_i = DR_i$ if $NS_i = \text{Yes}$.

3. Node i's participation in contending for access to the channel is dependent on the rate at which node i generates RTS messages and if it is time to expect activity from node i. The probability factor should have the following properties:

   - If it is not time to expect activity from node i and node i generates RTS messages slowly, node i does not want to participate in contending for access to the channel.
   - It is uncertain if node i wants to participate in contending for access to the channel if it generates RTS messages slowly and it is time to expect activity from the node.
   - Node i is slightly more likely to not participate in contending for access to the channel if the rate at which node i generates RTS messages is high but it is not yet time to expect activity from the node.
   - Node i wants to participate in contending for access to the channel if it generates RTS messages quickly and it is is time to expect activity from the node.

$$CP_i = \begin{cases} \text{No} & \text{if } RGR_i = \text{Low};\ EX_i = \text{No} \\ ? & \text{if } RGR_i = \text{Low};\ EX_i = \text{Yes} \\ \pm\text{No} & \text{if } RGR_i = \text{High};\ EX_i = \text{No} \\ \text{Yes} & \text{if } RGR_i = \text{High};\ EX_i = \text{Yes} \end{cases} \tag{5.7}$$

This logic results in the heuristic probability factor found in Table 5.4.

RGR_i	EX_i	CP_i	$\rho(CP_i \mid RGR_i, EX_i)$
Low	No	No	1.0
Low	No	Yes	0.0
Low	Yes	No	0.5
Low	Yes	Yes	0.5
High	No	No	0.6
High	No	Yes	0.4
High	Yes	No	0.0
High	Yes	Yes	1.0

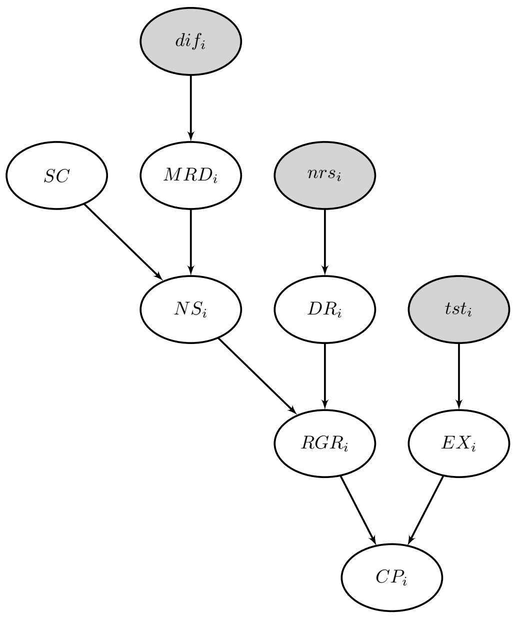
Table 5.4: Probability factor $\rho(CP_i|RGR_i, EX_i)$, using probabilities to describe how RGR_i and EX_i influence CP_i. Further details can be found in the main text and equation (5.7).

5.3.1.3 Calculating Probabilities

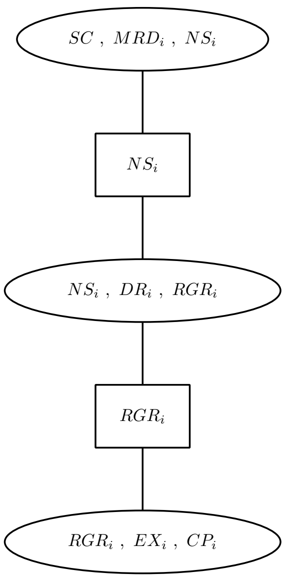
For each active neighbouring node, we are required to estimate the probability that the node wants to participate in contending for access to the channel. In order to illustrate the statistical dependencies between RVs (found in the probability factors), a Bayes network is used for each neighbouring node (Fig. 5.4a).

A cluster graph (Fig. 5.4b) for each neighbouring node i is created (from the probability factors), and belief propagation is used to calculate the probability that node i wants to participate in contending for access to the channel. Note that the cluster graph is a junction tree as it contains no loops, resulting in quick and accurate calculations for the probabilities.

The total number of nodes contending for access to the channel equals the sum of all the individual contending nodes -- in this case, a sum of RVs. Summing RVs results in a convolution of the probability densities of the RVs (proof in Appendix B). Thus to estimate the total number of nodes contending for access to the channel, the probability densities of all the nodes' CP_i are convoluted and the number of nodes with the highest probability are chosen.



(a) Bayes network illustrating the statistical dependencies between the RVs of the cluster graph augmented CSMA/CA MAC protocol; shading the observed RVs.



(b) Cluster graph used to calculate the probability that a neighbouring node wants to transmit data (for the cluster graph augmented CSMA/CA MAC protocol). Note that the cluster graph contains no loops, thus it is a junction tree, resulting in quick and accurate calculations.

Figure 5.4

5.3.2 Choosing the size of the contention window

The estimated number of nodes contending for access to the channel can be used to determine the optimal size of the contention window (CW). Since there are complex interactions between nodes contending for access to the channel, simulations were performed to determine the correlation between the number of contending nodes and the optimal size for CW, simulating various numbers of contending nodes for different values of a fixed CW. The optimal size for CW corresponds with the value of CW -- which results in the maximum channel efficiency, as maximum channel efficiency corresponds with the least time wasted due to nodes contending for access to the channel.

Fig. 5.5 illustrates the optimal size of CW versus the number of contending nodes. A linear correlation between the number of contending nodes and the value of CW is assumed. Note that a slightly bigger CW was chosen to increase the probability that a node wins access to the channel and to accommodate for nodes becoming active. The disadvantage of a slightly bigger CW is minimal, as it increases the delay of data packets insignificantly and reduces efficient use of the channel minimally.

Note that the linear correlation between the number of contending nodes and the size of CW is an optimal decision strategy, stating the optimal choice for CW given the number of contending nodes. It is thus not necessary to use a decision cluster graph to assign a decision strategy.

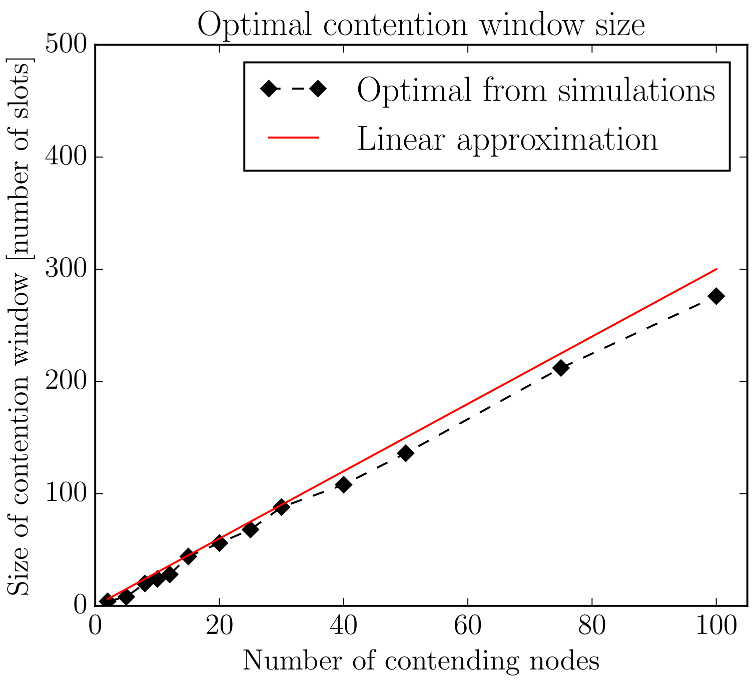


Figure 5.5: Optimal contention window size versus number of contending nodes. Note that the linear approximation has a slightly bigger CW to increase the probability that a node wins access to the channel and to accommodate for nodes becoming active.

5.4 Conclusions and Recommendations

The following observations and recommendations can be made about our method:

- The cluster graph used to calculate the probability that a node wants to participate in contending for access to the channel is a simple JT, resulting in an efficient method for calculating the probabilities.
- It is possible to create a more accurate and complex model, and one way this can be achieved is by creating a Markov chain, modelling the possibilities as time advances. However, it is important to remember that a more complex

model could result in extended calculation times, degrading performance more than it improves it.

- We compensate for inaccurate estimations (due to the simplicity of our model) by choosing a slightly bigger CW. The disadvantages of a slightly bigger CW are minimal, increasing the latency of packets and reducing channel efficiency only slightly.

Chapter 6

Experiments and Results

The main objective of this work is to improve the performance (and demonstrate the feasibility) of our PGM-augmented MAC protocols. In order to verify whether augmenting current MAC protocols with PGMs improves performance sufficiently to justify the extra calculations, a number of experiments were performed. These experiments were performed at the best- and worse-case conditions for the specific protocol; this was done in order to check whether the augmented protocol maintains the performance for the best-case scenario and if it improves the worse-case scenario. Results were obtained by simulating a network with the WSNet[1], an event-driven simulator for wireless networks.

6.1 Round Robin MAC Protocol

The Round Robin MAC protocol is efficient under high traffic loads, since there are no collisions in the network. However, the protocol is inefficient at low traffic loads; a node with data to transmit has to wait to be polled, increasing the transmission delay (latency) of data packets. We therefore want to determine whether our PGM-augmented methods maintains this performance and fairness when the network is under heavy equal-traffic loads and if our methods reduce latency when the network is under an unbalanced traffic load (where a few nodes have a significantly higher data rate than the others). Thus we performed 2 types of experiments for the Round Robin protocol:

1. An equal traffic load experiment; all nodes have an equal (and high) data rate.

2. An unequal traffic load experiment; a few nodes have a high data rate while the rest have a low data rate.

[1]WSNet simulator available at: http://wsnet.gforge.inria.fr

6.1.1 Experimental Set-up

The experiments for the Round Robin protocol simulations were set up as follows:

- There are 30 nodes in the network, 1 master node and 29 slave nodes, all within transmission range of one another.
- Slave nodes generate data packets according to a Poisson distribution, generating λ packets every 10ms.
- The destination node of all the packets is the master node.
- Each experiment is simulated 100 times to account for the randomness involved in generating packets.
- 4 MAC protocols were simulated:
  1. The normal Round Robin MAC protocol, polling slave nodes sequentially.
  2. The cluster graph augmented MAC protocol, deciding which slave node to poll based on probabilities calculated with a cluster graph (method of Section 4.3).
  3. The decision cluster graph augmented MAC protocol, using a decision cluster graph to decide which slave node to poll (decision cluster graph method of Section 4.2).
  4. The strong JT augmented MAC protocol, using a strong JT to decide which slave node to poll (strong JT method of Section 4.2).

The only difference between the two experiments is that for the equal load experiment, all slave nodes have the same λ, while in the unequal load experiment, a few slave nodes have a significantly higher λ than the the rest of the nodes. Additional simulation parameters can be found in Appendix C, Table C.1.

The following statistics were recorded during the simulations in order to evaluate the performance of the protocols:

- Total number of packets reaching their destination.
- Average latency of the data packets.
- Number of polls each slave node received.

In order to evaluate performance we consider effective channel usage and average latency:

- channel efficiency $= \frac{\text{time spent sending payloads}}{\text{total simulated time}}$

- A packet's latency is calculated as the time difference between when it was generated and when it arrived at its destination.

To gauge the fairness in the network we consider the number of polls that each individual node received:

- Minimum number of polls that any slave node received.
- Maximum number of polls that any slave node received.

6.1.2 Results and Conclusions

The results for evaluating the performance of the equal traffic load experiment are illustrated in Fig. 6.1 (illustrating channel efficiency) and Fig. 6.2 (illustrating the average latency). From Fig. 6.1, the effective channel usage for the different protocols are almost exactly equal, showing that the augmented protocols do not compromise channel efficiency. However, when considering the average latency for the different protocols (Fig. 6.2), the latencies of the augmented protocols are higher. The reason for this increased latency becomes clear when we consider the number of polls the slave nodes received. The discrepancy between the channel efficiency and average latency is possible because while there was no packet loss in these simulations, some nodes had to wait longer to be polled, increasing the latency of their data packets. From these results, we can conclude that the augmented protocols remain efficient under high traffic loads, maintaining efficient channel usage. The decision cluster graph and the strong JT augmented protocol show similar results, illustrating that decision cluster graphs are viable.

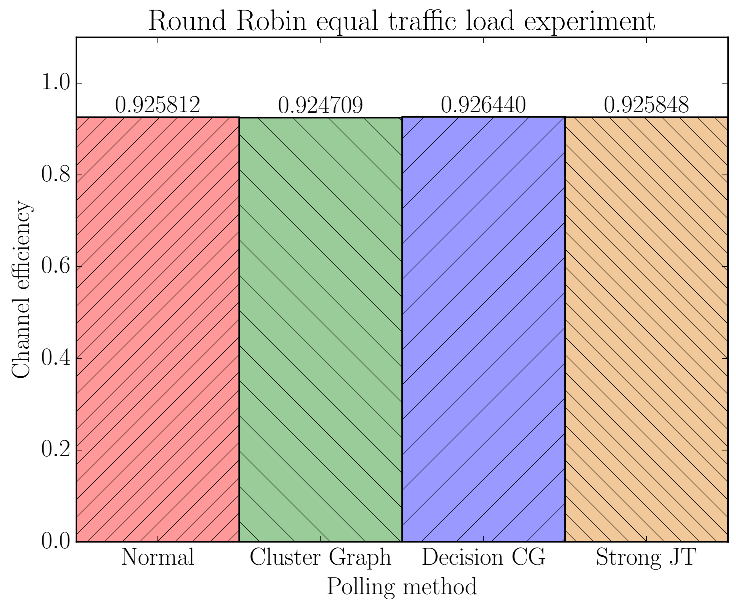


Figure 6.1: Channel efficiency for the Round Robin equal traffic load experiment. Note that the channel efficiency for the different protocols are equal.

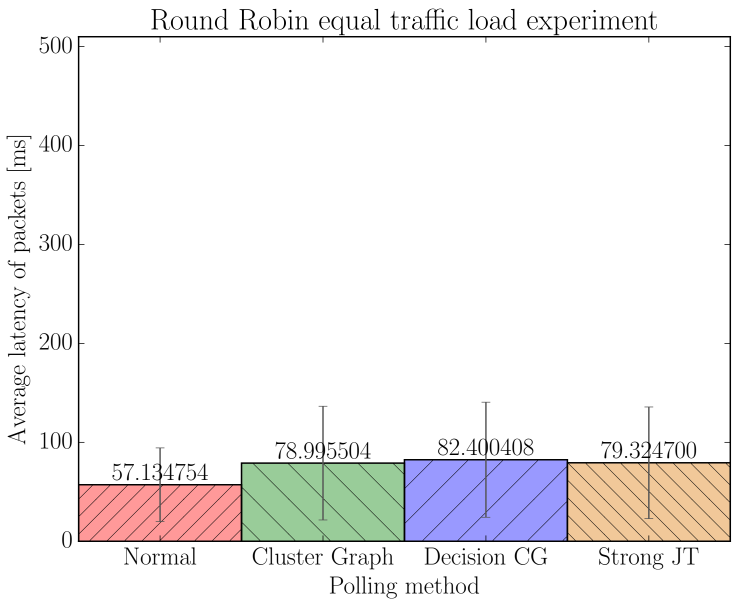


Figure 6.2: Average latency for the Round Robin equal traffic load experiment. Note that the average latency of the augmented protocols are higher, showing a slight decrease in performance.

The results of evaluating the fairness in the equal traffic load experiment are illustrated in Fig. 6.3 (illustrating the minimum and maximum number of polls received by slave nodes). Note that the normal MAC protocol is always fair, since it polls nodes in sequence, making it an excellent benchmark. This result is observed in Fig. 6.3, since the minimum and maximum number of polls that a slave node received are equal. The augmented protocols compromise fairness slightly, since there is some difference between the minimum and maximum number of polls a slave node received. The decision cluster graph augmented protocol has the biggest difference, compromising fairness even more. However, the minimum number of polls received by slave nodes is never too small, and thus the protocols remain reasonably fair.

The reason for the increased latency of data packets in the augmented protocols now becomes evident: some nodes are polled more often, decreasing the latency of packets from these nodes, while other nodes are polled less often, increasing the latency of packets from these nodes. The decrease is less than the increase, resulting in an increase in the overall average latency of packets. This unequal polling of slave nodes is the result of inaccurate estimations of the probabilities and utilities of RVs in the network, especially the data rate of a node. The estimated data rate of a node depends on how long the node had to wait to be polled and the number of packets it then transmitted. If a node is polled too quickly with respect to its actual data rate, it might have an empty (or very short) packet queue and the system can estimate that it has a low data rate. Thus the system would wait longer to poll this node, resulting in estimating a data rate that is too high, repeating the process. The probability factors are set up to minimize this effect, but the fact that the input variables are discretized further adds to this inaccuracy, since some information is lost resulting in inaccurate probabilities. Furthermore, the inaccuracy of probabilities and utilities introduced by loopy decision cluster graphs worsens the estimations and results in larger differences in the number of polls that slave nodes receive, as illustrated in Fig. 6.3.

However, this inaccuracy in estimating the data rate is most prominent when most nodes have an equal and high data rate, since as soon as the system differentiates between equal nodes, the differentiation keeps happening. In the case where there is a significant distinction between the data rates of nodes (which the system can easily identify), this inaccuracy is not manifested. The inaccuracy can be reduced by creating a more complex (and thus more accurate) model. Since the node data rates differ for most practical implementations, we did not optimize the protocols for this scenario. Furthermore, the objective of this project is to demonstrate the feasibility of a PGM-augmented protocol and not to maximize performance to the limit, and thus the creation of a more accurate model is left for future work.

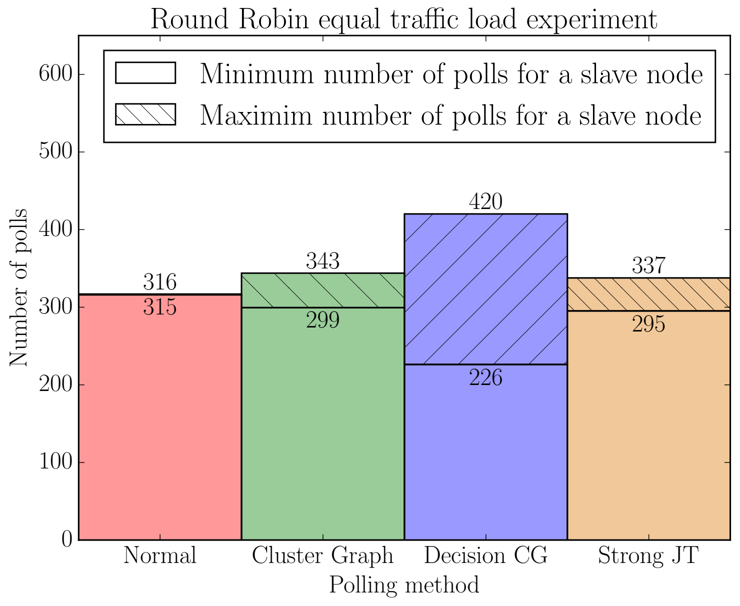


Figure 6.3: Number of polls slave nodes received for the Round Robin equal traffic load experiment. Note that there is no difference in the minimum and maximum number of polls received by slave nodes for the normal protocol; there is a small difference in the minimum and maximum number of polls received by slave nodes for the cluster graph and strong JT augmented protocol; and that the decision cluster graph augmented protocol has the biggest difference in the minimum and maximum number of polls received by slave nodes. Thus the decision cluster graph augmented protocol is slightly less fair than the other protocols for this experiment as all nodes have the same data rate.

The results for evaluating the performance of the unequal traffic load experiment are illustrated in Fig. 6.4 (illustrating channel efficiency) and Fig. 6.5 (illustrating the average latency). Although the channel efficiency of the different protocols is similar, the average latency of data packets of the augmented protocols is significantly lower than the standard protocol. This discrepancy between channel efficiency and average latency is possible as there was no packet-loss in these simulations. The standard protocol wastes time polling nodes with no data to transmit and thus the nodes without packets to transmit wait longer to transmit their data packets, increasing the latency. The reason for this improvement can be explained when we consider the number of polls that slave nodes receive, illustrated in Fig. 6.6. From this figure it can be observed that there is a significant difference between the minimum and maximum number of polls that slave nodes receive for the augmented protocols, indicating that the slave nodes with a higher data rate get polled more often. When nodes that have a higher data rate are polled more often, the latency decreases as less time is wasted polling nodes with no data to transmit, which in turn allows the nodes that do have data to transmit to do so more often. These results show that the estimations of all the augmented protocols are quite accurate. The results also show that the decision cluster graph augmented protocol gives similar results to the strong JT, illustrating that decision cluster graphs can be used to solve practical decision-making tasks effectively.

Gauging the fairness of the protocols for the unequal traffic load experiment is slightly more subjective. We consider the minimum number of polls that slave nodes received (illustrated in Fig. 6.6). If a node did not receive enough polls, it indicates that the node was disregarded by the master node, resulting in an unfair protocol. As expected, the minimum number of polls a slave node received for the augmented protocols is lower than the standard protocol, but not low enough to be seen as being disregarded by the master node; thus the augmented protocols maintain reasonable fairness in the network.

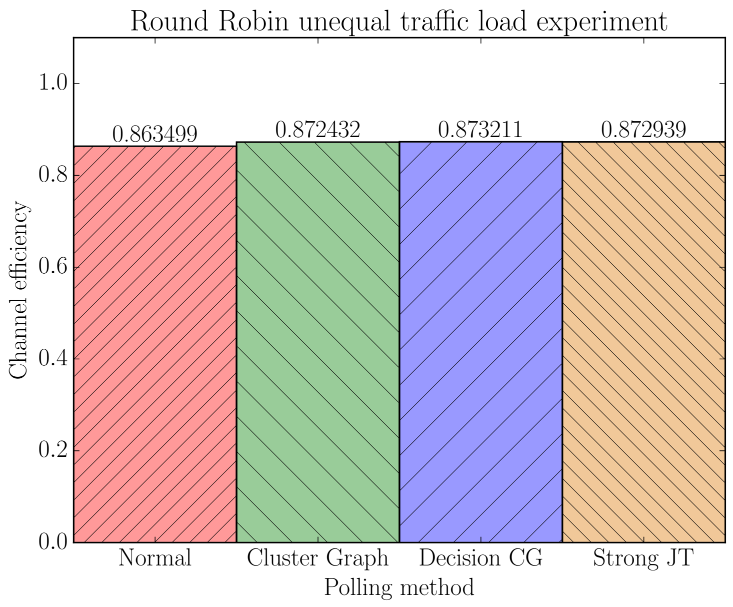


Figure 6.4: Channel efficiency for the Round Robin unequal traffic load experiment. Note that the channel efficiency for the different protocols are equal.

From these results we can conclude that augmenting the Round Robin MAC protocol with a PGM is beneficial, decreasing the latency of packets when the network is under an unequal traffic load, while maintaining efficiency and a reasonably fair protocol under heavy traffic loads.

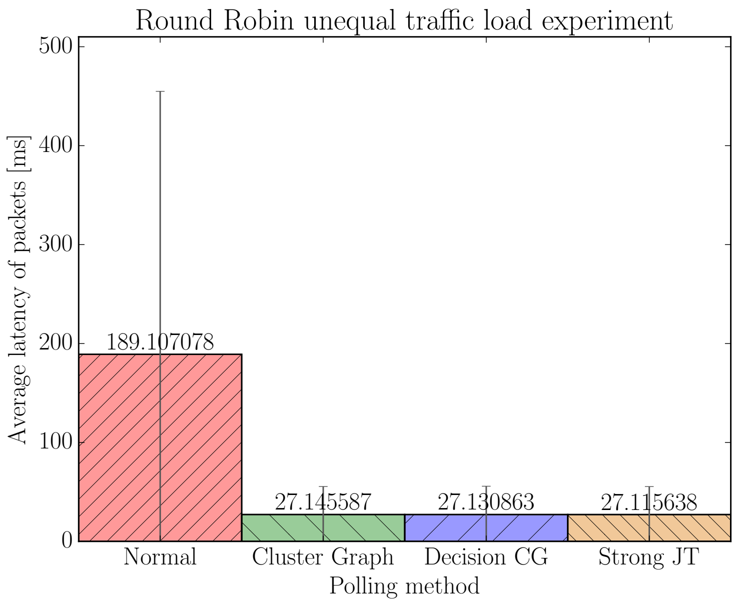


Figure 6.5: Average latency for the Round Robin unequal traffic load experiment. Note that the augmented Round Robin MAC protocols have a significantly lower average latency than the normal Round Robin MAC protocol.

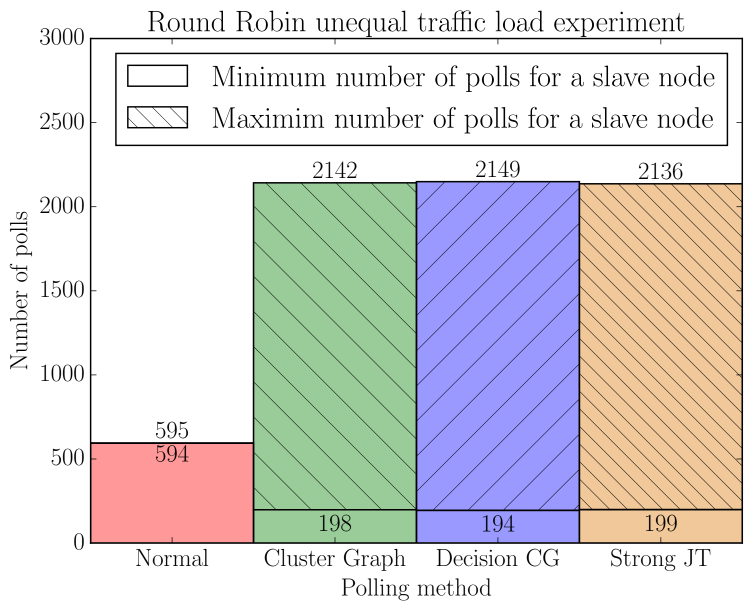


Figure 6.6: Number of polls slave nodes received for the Round Robin unequal traffic load experiment. Note that the augmented protocols have a significant difference in the minimum and maximum number of polls received by slave nodes. Since not all slave nodes have the same data rate, this difference show that nodes with a higher data rate are polled more often, resulting in the improved performance (lower latency) found in Fig. 6.5. Also note that the minimum number of polls which a slave node received is not low enough to be seen as being disregarded by the master node.

6.2 CSMA/CA MAC Protocol

The CSMA/CA MAC protocol is efficient at low traffic loads as a node would be granted access to the medium almost immediately, resulting in a very short delay. However, the protocol is inefficient at high traffic loads as collisions can occur frequently, obstructing data transmission. Therefore, our objective is to determine whether our PGM-augmented protocol improves channel efficiency when the network is under a heavy traffic load, and if it maintains low latencies during low traffic loads for a typical use case. Thus we performed 2 types of experiments for the CSMA/CA protocol:

1. A saturation experiment, simulating a network under the maximum stable traffic load.
2. A Poisson experiment, generating data packets according to a Poisson distribution, closer to a typical use case.

6.2.1 Experimental Set-up

The saturation experiment for the CSMA/CA protocol simulations were set up as follows:

- There are N nodes (a variable parameter) in the network, all within transmission range of each other.
- Nodes always have a data packet to transmit; as soon as the current packet reaches its destination, a new packet is generated. This maximizes the traffic load while keeping the network in a stable condition.
- A node may contend for channel access an unlimited number of tries per packet.

The Poisson experiment for the CSMA/CA protocol simulations was set up as follows:

- There are $N = 25$ nodes in the network, all within transmission range of each other.
- Nodes generate data packets according to a Poisson distribution, generating λ packets (a variable parameter) every 10ms.
- A data packet is dropped after 7 attempts of contending for access to the channel.

Experimental set-up conditions that are common between the two experiments:

- The destination node for the data packets is a random node in the network.
- Each experiment was simulated 100 times to account for the randomness involved in the protocol and in generating packets (for the Poisson experiment).
- The following CSMA/CA MAC protocols were simulated:
  1. The standard CSMA/CA MAC protocol with parameter choices: $CW_{min} = 3$ and $CW_{max} = 127$.
  2. The standard CSMA/CA MAC protocol with parameter choices: $CW_{min} = 31$ and $CW_{max} = 1023$.
  3. The standard CSMA/CA MAC protocol with parameter choices: $CW_{min} = 255$ and $CW_{max} = 8191$.
  4. The cluster graph augmented CSMA/CA MAC protocol (method of Chapter 5).

Additional simulation parameters can be found in Appendix C, Table C.2.

The following statistics were recorded during the simulations in order to evaluate the performance of the methods:

- Total number of packets reaching their destination (also logging the number of packets for individual nodes).
- Average latency of the data packets.

In order to evaluate the performance of the protocol, we consider effective channel usage and average latency:

- channel efficiency $= \frac{\text{time spent sending payloads}}{\text{total simulated time}}$
- The latency of a packet is calculated as the time difference between when it was generated and when it arrived at its destination.

To gauge the fairness of the protocols, we consider the distribution (illustrated as a whisker-diagram) of the effective throughput ($\frac{\text{total payload deliverd [Mbits]}}{\text{total simulated time [s]}}$) of individual nodes, plotting the minimum, 25th percentile, median, 75th percentile, and maximum throughput of individual nodes.

6.2.2 Results and Conclusions

The results of evaluating the performance of the saturation experiment are illustrated in Fig. 6.7, illustrating channel efficiency. The channel efficiency of the cluster graph augmented protocol is basically as high as the standard protocol (Fig. 6.7), regardless of the parameter choice for the standard protocol. The cluster graph augmented protocol therefore adapts better to the size (and load) of the network to maintain efficient channel usage. The cluster graph augmented protocol is able to adapt better thanks to estimating the number of contending nodes and adjusting its contention window accordingly. There is little room for improvement here, as the augmented protocol comes close to the maximum channel efficiency of the standard protocol. Further improvement can be achieved by estimating the number of contending nodes more accurately with a more complex model. Since we focus on feasibility, it is beyond the scope of this work and is recommended for future work.

Fig. 6.8 illustrates the effective throughput distribution of the individual nodes as a whisker diagram (for the saturation experiment). The more compressed the whisker diagram, the fairer the method, since the throughput of the nodes is more evenly distributed. In this figure, for the $N = 20$ case, the whisker diagram for the standard protocol with the biggest contention window is the most compressed and thus the fairest of all. However, when looking at the channel efficiency for the corresponding number of transmitting nodes (Fig. 6.7), it under-utilizes the channel. The reason for this fairness is that the contention window is too big. Thus there is no (or minimal) collisions, giving all nodes a fair and equal chance to transmit. However, since the contention window is too big, a great deal of time is wasted with the extended contention period, resulting in inefficient channel usage. For all the other scenarios, the cluster graph augmented protocol has the most compressed whisker diagram; thus the protocol maintains (and in some cases improves) fairness in the network.

Note that the whisker diagrams for the standard protocol with contention windows that are too small is slightly skewed to higher individual throughput. This indicates that one or a few nodes continuously gain access to the channel, resulting in a slightly unfair protocol. The reason for this is that a contention window that is too small results in many collisions, which doubles the value of CW, and, once a node is successful at contending for access to the channel, CW is set to CW_{min} while the other nodes' CW remains big. Thus the successful node is more likely to be granted access to the channel sooner as it can only choose from earlier time slots.

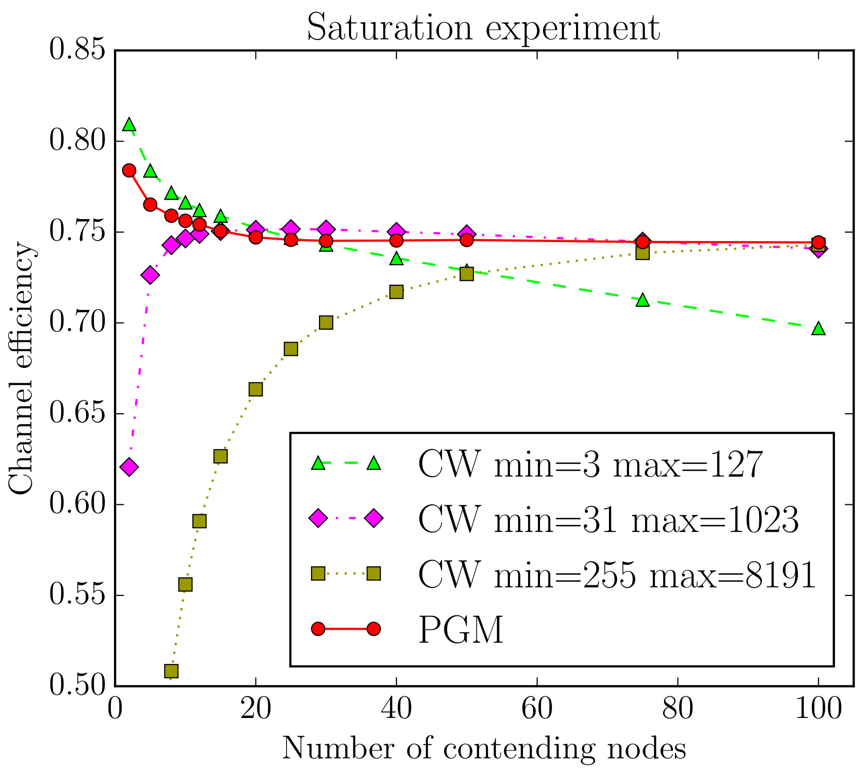


Figure 6.7: Channel efficiency for the CSMA/CA saturation experiment. Note that the PGM-augmented CSMA/CA MAC protocol maintains an efficient channel usage regardless of the number of contending nodes, while the efficiency of the normal CSMA/CA MAC protocol diminishes depending on the number of contending nodes and the parameter choices.

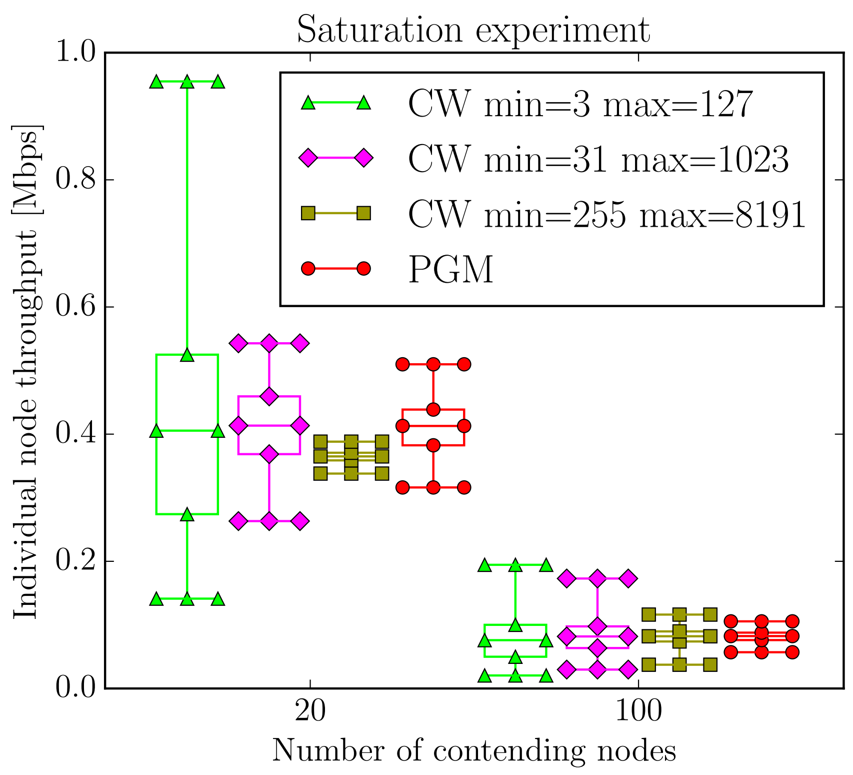


Figure 6.8: Distribution of individual node throughput for the CSMA/CA saturation experiment illustrated as a whisker-diagram. Note that the PGM-augmented CSMA/CA MAC protocol has the most compressed whisker-diagram, except for the normal protocol with the biggest contention window in the $N = 20$ case. However, from Fig. 6.7, the corresponding channel efficiency for this normal protocol is worse. Thus the PGM-augmented CSMA/CA MAC protocol maintains a fair network.

The results of evaluating the performance of the Poisson experiment are illustrated in Fig. 6.9 (illustrating channel efficiency) and Fig. 6.10 (illustrating average latency). From Fig. 6.9, the channel efficiency of the cluster graph augmented protocol is as high as the standard protocol with optimal parameter choices. From Fig. 6.10, the average latency of data packets is basically as low as the standard protocol with optimal parameter choices. Thus the cluster graph augmented protocol maintains the low latency and efficient channel usage of the standard CSMA/CA MAC protocol for a typical use case.

From Fig. 6.10, it is observed that the average latency increases dramatically when the network becomes congested (when the data rate of the nodes exceeds the capacity of the network). The reason for this is that data packets are generated faster than they can be transmitted, resulting in numerous packets in the transmission queue waiting to be transmitted, increasing their latency significantly.

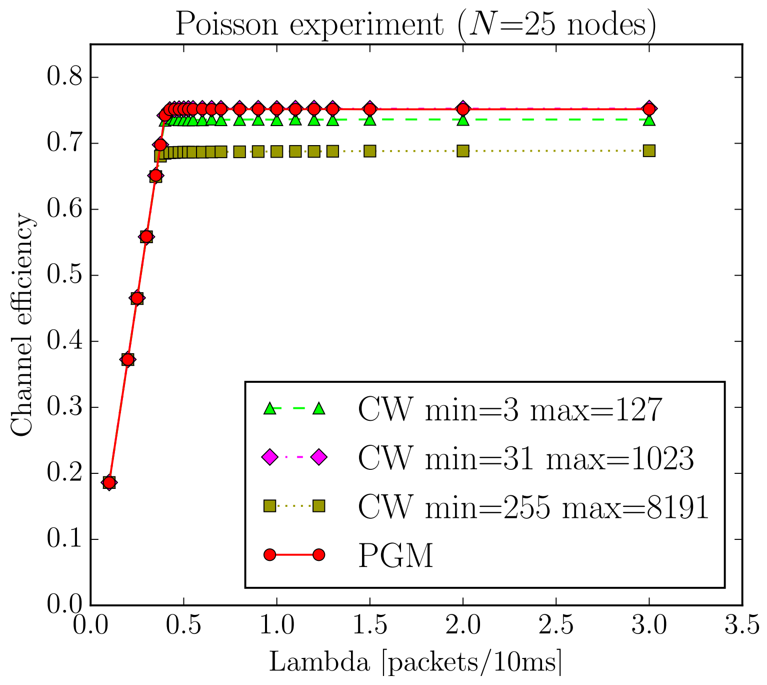


Figure 6.9: Channel efficiency for the CSMA/CA Poisson experiment (25 nodes). Note that the channel efficiency for the PGM-augmented protocol is similar to the normal protocol with the highest efficiency. Also, the channel efficiency saturates at $\lambda \approx 0.45$ packets/10ms and the network becomes congested at higher data rates.

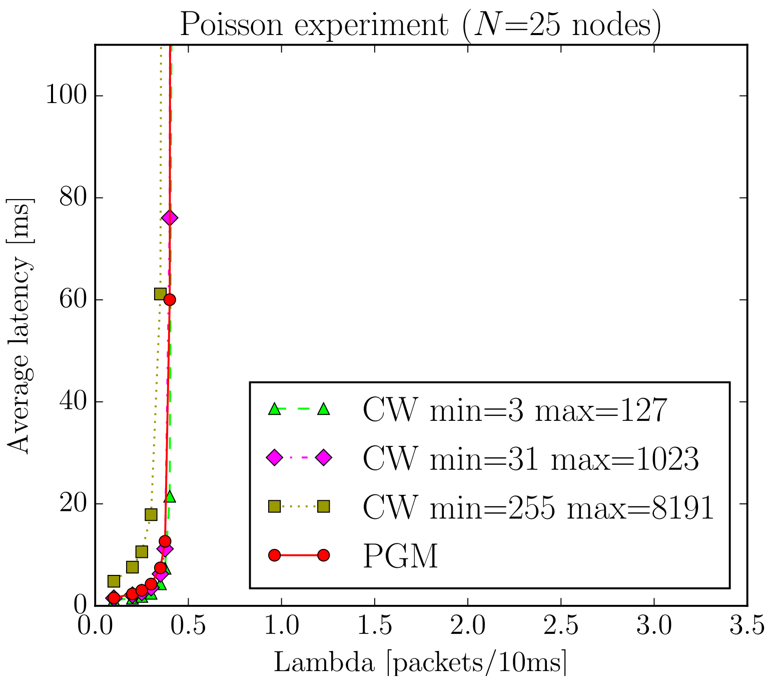


Figure 6.10: Average latency for the CSMA/CA Poisson experiment (25 nodes). Note that the average latency for the PGM-augmented protocol is similar to the normal protocol with the lowest average latency, thus low latencies at low loads are maintained. Also note that the average latency increases significantly when the network becomes congested.

The results of evaluating the fairness in the Poisson experiment are illustrated in Fig. 6.11, illustrating the throughput of individual nodes as a whisker diagram. From this figure, the whisker diagram for all the protocols is equally compressed when the traffic load is low thanks to an extremely successful contention period (only one or a few nodes are contending for access to the channel). However, when the network becomes congested, the standard protocol with a big CW has the most compressed whisker diagram and is thus the fairest, but the corresponding efficient channel usage is the worse. The cluster graph augmented protocol has the second-most compressed whisker diagram while also maintaining efficient usage of the channel. Thus the cluster graph augmented protocol maintains (or improves) fairness in the Poisson experiments.

Note that for the Poisson experiments, the whisker diagrams of the standard protocol are slightly skewed to lower individual throughput. The reason it differs from the saturation experiments (where it is skewed to higher throughput) is that a node is allowed to drop a packet when contention for access to the channel failed too many times. This only happens to a few nodes, as it effectively lowers the overall data rate of the network, allowing other nodes to be slightly more successful while decreasing its own data rate and thus throughput.

From these results we can conclude that augmenting the CSMA/CA MAC protocol with a PGM is beneficial, improving channel efficiency regardless of the traffic load in the network, while maintaining a fair network. It also maintains low latencies for a typical use case.

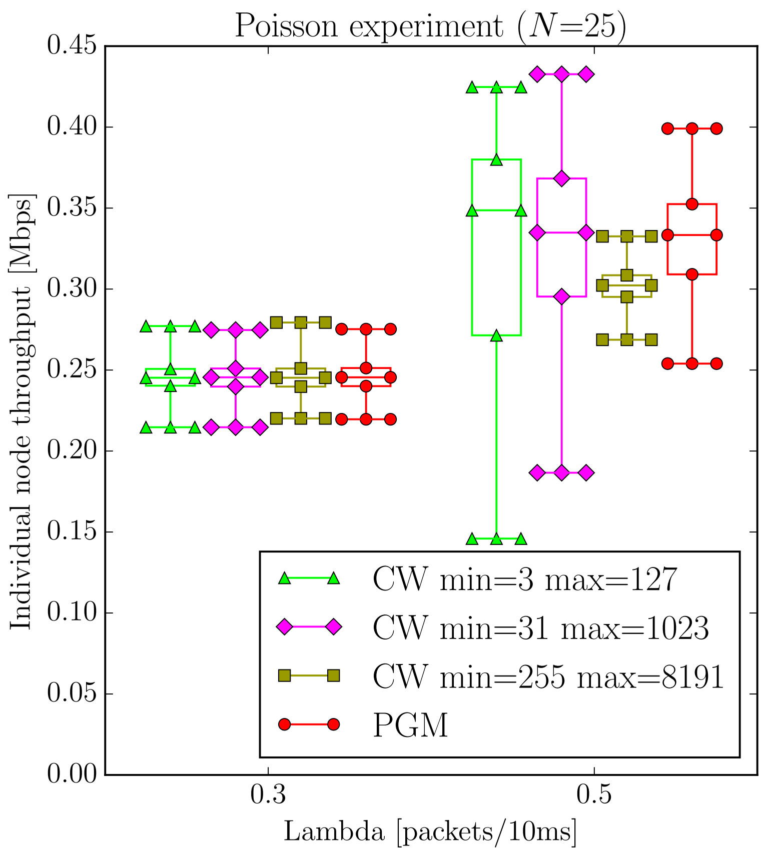


Figure 6.11: Distribution of individual node throughput for the CSMA/CA Poisson experiment (25 nodes) as a whisker-diagram. Note that when the traffic load is low, the whisker-diagram for the different protocols are equally compressed, since contention is happening quickly and successfully, resulting in a fair network. Also, when the data rate is higher, the results are similar to the saturation experiment of Fig. 6.8; the PGM-augmented protocol has the most compressed whisker-diagram, except when the contention window is too big and compromises channel efficiency (seen in Fig. 6.9). Thus the PGM-augmented CSMA/CA MAC protocol maintains a fair network.

Chapter 7

Conclusions and Recommendations

7.1 Decision Cluster Graphs

The following conclusions and recommendations for decision cluster graphs are made:

- We expanded the current theory of the decision-making task by defining the normalization operation for decision potentials. This allows for creating (loopy) decision cluster graphs as opposed to strong junction trees (JTs), as the former have been demonstrated to offer a more efficient method of calculating probabilities and utilities, as well as assigning decision strategies.

- Similarly to normal cluster graphs, loopy decision cluster graphs lead to imprecise probabilities and utilities, which can result in sub-optimal decision strategies. The issues (and recommendations) we identified with loopy decision cluster graphs are as follows:

  1. The independence assumption made between random variables (RVs) (Section 3.3) can lead to inaccurate probabilities and utilities, resulting in sup-optimal decision strategies. Thus be wary of RVs with strong statistical dependencies, particularly when the cluster graph is loopy.
  2. The effect of decision strategies that do not rely on the input variables of the decision variable (Section 3.4) can be minimized by observing the input variables of the system or, if possible, structuring the problem in such a way that decision variables are not directly dependent on other variables. However, both these recommendations require that the method should be performed at each decision in time.
  3. Cyclic relevance graphs result in sub-optimal decision strategies when the decision strategies are optimized one at a time (even for strong JT). The reason for this is that the optimal strategies are reliant on each

other and should, if possible, be optimized together for global optimal strategies.

- The results of the loopy decision cluster graph and strong junction tree approaches for determining which node to poll in the Round Robin MAC protocol are reasonably similar; thus we conclude that a loopy decision cluster graph can be used to solve practical problems. However, when considering the number of polls that nodes received during the equal traffic load experiment, there are some discrepancies between the strong JT and the loopy decision cluster graph. This indicates that, similar to the behaviour of loopy cluster graphs (upon which decision cluster graphs are based), the results can be slightly inaccurate, leading to slightly sub-optimal decision strategies.
- For future work regarding decision cluster graphs, we recommend testing and validating the method for general decision-making problems. We suspect similar behaviour to normal cluster graphs: the applicability and performance will depend on the problem to be solved.

7.2 Augmented Round Robin protocols

The conclusions and recommendations for augmenting the Round Robin MAC protocol with probabilistic graphical models (PGMs) are listed below.

- We created 3 augmented Round Robin MAC protocols:
    1. Decision cluster graph augmented Round Robin MAC protocol, using a loopy decision cluster graph to determine which slave node to poll next.
    2. Strong JT augmented Round Robin MAC protocol, using a strong JT (created from the loopy decision cluster graph of the previous method) to determine which slave node to poll next.
    3. Cluster graph augmented Round Robin MAC protocol, using a cluster graph to estimate probabilities and deciding which node to poll based on the calculated probabilities.
- All the augmented methods perform similarly and improve the standard Round Robin MAC protocol, maintaining an efficient protocol under high traffic loads and reducing latency under unequal traffic loads. However, our methods are not perfect as some nodes are polled slightly more often than others when they have the same data rate. The reason for this is that our estimations are not always exact. Since the decision cluster graph method is loopy, its estimations are the least precise of the 3. These results show that estimating uncertainties in a network with the help of a PGM (even a simple one) and adapting the MAC protocol accordingly is beneficial.

- In the Round Robin MAC protocol, the master node is the only node that is required to implement a PGM. Thus a more complex (and more accurate) model can be practical, since the hardware of the slave nodes can remain extremely simple. The following modifications are suggested for a more complex and accurate model:

  - Instead of discretizing observed variables, we suggest using functions to determine the probabilities of variables that are dependant on the observed variables (similar to our approach in the augmented CSMA/CA protocol). This minimizes information loss and allows for better differentiation between nodes.
  - Improve the accuracy of the model by creating a more complex model and logging information about the network over a longer time. We suggest implementing a Markov chain for a more complex model, as it will improve estimations for transient effects. Since transmissions are usually two-way communication, modelling interactions between nodes can also improve performance.
  - Machine-learning techniques can be used to optimize the probability factors in order to maximize performance.

  For a more complex model, it is necessary to be wary of RVs that have strong statistical dependencies as they could result in highly inaccurate calculations, particularly if the cluster graph is loopy. However, a simple model should not be underestimated, as it usually results in quick calculations and thus minimal overhead. Therefore, comparing the results of a simple model to a more complex model in a physical experiment is also recommended for future work.

- Creating a hybrid protocol of a Round Robin and Binary Tree MAC protocol is also recommended for future work, where the master node is allowed to poll groups of nodes and a PGM is used to determine which group to poll. The advantage of such a protocol is that during low traffic loads a bigger group can be polled, resulting in short transmission delays since nodes receive a poll intended for them more often. In addition, when the traffic load is heavy, the master node polls individual nodes, conserving the effectiveness of the Round Robin MAC during heavy traffic loads.

7.3 Augmented CSMA/CA protocol

The following conclusions and recommendations for augmenting the CSMA/CA MAC protocol with a PGM are made:

- For the CSMA/CA MAC protocol, we implemented a cluster graph to estimate the number of nodes participating in contention for access to the channel. From this estimation, we adapted the length of the contention window. Although we used a reasonably simple cluster graph, our method performed quite well, adapting to the size and load of the network to produce an efficient and fair network protocol.

- Although our method performed quite well, it did not reach the maximum realistic performance. This is recommended for future work and we give the following suggestions to improve upon our method:

  - Create a more complex model for better estimations for RVs. For this we suggest implementing a Markov chain to better estimate transient effects. With a better estimation for RVs, the length of the contention window can be adjusted more accurately, creating a more effective protocol. Since transmissions are usually two-way communication, modelling interactions between nodes can also improve performance.
  - Use machine-learning techniques to optimize the probability factors in order to maximize performance.

  Similar to our recommendations for an augmented Round Robin MAC protocol, when using a more complex model for an augmented CSMA/CA MAC protocol, it is necessary to be wary of RVs that have strong statistical dependencies. This is because they could result in highly inaccurate calculations, particularly if the cluster graph is loopy. However, a simple model should not be underestimated, as it usually results in quick calculations and thus minimal overhead. Comparing the results of a simple model to a more complex model in a physical experiment is therefore also recommended for future work.

Appendices

Appendix A

Normalization proofs

We prove that that normalization of decision potentials (equation (3.4)) satisfies both equation (3.5) and (3.6).

Reference equations for decision potential operations:

Absorb, equation (3.1):

$$\gamma_1 \oplus \gamma_2 = (\rho_1 \cdot \rho_2 \ , \ \mu_1 + \mu_2)$$

Normalize, equation (3.4):

$$norm(\gamma) = \left(\frac{\rho}{\sum \rho} \ , \ \mu - \frac{\sum \rho \cdot \mu}{\sum \rho} \right)$$

Theorem 1. *$norm(\gamma) = norm(norm(\gamma))$* (equation (3.5))

Proof. Define:

$$\begin{aligned} S_\rho &= \sum \rho \\ S_{\rho\mu} &= \sum \rho \cdot \mu \\ \rho' &= \frac{\rho}{S_\rho} \\ \mu' &= \mu - \frac{S_{\rho\mu}}{S_\rho} \end{aligned}$$

Proof:

$$\begin{aligned} LHS &= norm(\gamma) \\ &= \left(\frac{\rho}{\sum \rho} \ , \ \mu - \frac{\sum \rho \cdot \mu}{\sum \rho} \right) \\ &= \left(\frac{\rho}{S_\rho} \ , \ \mu - \frac{S_{\rho\mu}}{S_\rho} \right) \\ &= (\rho' \ , \ \mu') \end{aligned}$$

$$
\begin{aligned}
RHS &= norm(norm(\gamma)) \\
&= norm\left((\rho' \ , \ \mu')\right) \\
&= \left(\frac{\rho'}{\sum \rho'} \ , \ \mu' - \frac{\sum \rho' \cdot \mu'}{\sum \rho'}\right) \\
\frac{\rho'}{\sum \rho'} &= \frac{\rho'}{\sum \frac{\rho}{S_\rho}} \\
&= \frac{\rho'}{\frac{1}{S_\rho} \cdot \sum \rho} \\
&= \frac{\rho'}{\frac{1}{S_\rho} \cdot S_\rho} \\
&= \rho' \\
\therefore \sum \rho' &= 1 \\
\mu' - \frac{\sum \rho' \cdot \mu'}{\sum \rho'} &= \mu' - \sum \rho' \cdot \mu' \\
&= \mu' - \sum \left(\frac{\rho}{S_\rho}\right) \cdot \left(\mu - \frac{S_{\rho\mu}}{S_\rho}\right) \\
&= \mu' - \frac{1}{S_\rho} \cdot \sum \left(\rho \cdot \mu - \rho \cdot \frac{S_{\rho\mu}}{S_\rho}\right) \\
&= \mu' - \frac{1}{S_\rho} \cdot \left(\sum \rho \cdot \mu - \sum \rho \cdot \frac{S_{\rho\mu}}{S_\rho}\right) \\
&= \mu' - \frac{1}{S_\rho} \cdot \left(S_{\rho\mu} - \frac{S_{\rho\mu}}{S_\rho} \cdot \sum \rho\right) \\
&= \mu' - \frac{S_{\rho\mu}}{S_\rho} \cdot \left(1 - \frac{1}{S_\rho} \cdot S_\rho\right) \\
&= \mu' - \frac{S_{\rho\mu}}{S_\rho} \cdot (1 - 1) \\
&= \mu' - \frac{S_{\rho\mu}}{S_\rho} \cdot 0 \\
&= \mu' \\
\therefore RHS &= (\rho' \ , \ \mu')
\end{aligned}
$$

$$\therefore LHS = RHS$$

$\square$

Theorem 2. *$norm(\gamma_1 \oplus \gamma_2) = norm(\gamma_1) \oplus norm(\gamma_2)$* (equation (3.6))

Proof.

$$\begin{aligned}
LHS &= norm(\gamma_1 \oplus \gamma_2) \\
&= norm\left((\rho_1 \cdot \rho_2\ ,\ \mu_1 + \mu_2)\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum\sum \rho_1 \cdot \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum\sum(\rho_1 \cdot \rho_2) \cdot (\mu_1 + \mu_2)}{\sum\sum \rho_1 \cdot \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum\sum(\rho_1 \cdot \rho_2) \cdot (\mu_1 + \mu_2)}{\sum \rho_1 \cdot \sum \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum\sum(\rho_1 \cdot \rho_2) \cdot \mu_1 + (\rho_1 \cdot \rho_2) \cdot \mu_2)}{\sum \rho_1 \cdot \sum \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum\sum \rho_1 \cdot \rho_2 \cdot \mu_1 + \sum\sum \rho_1 \cdot \rho_2 \cdot \mu_2}{\sum \rho_1 \cdot \sum \rho_2}\right)
\end{aligned}$$

$$\begin{aligned}
RHS &= norm(\gamma_1) \oplus norm(\gamma_2) \\
&= \left(\frac{\rho_1}{\sum \rho_1}\ ,\ \mu_1 - \frac{\sum \rho_1 \cdot \mu_1}{\sum \rho_1}\right) \oplus \left(\frac{\rho_2}{\sum \rho_2}, \mu_2 - \frac{\sum \rho_2 \cdot \mu_2}{\sum \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 - \frac{\sum \rho_1 \cdot \mu_1}{\sum \rho_1} + \mu_2 - \frac{\sum \rho_2 \cdot \mu_2}{\sum \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum \rho_2 \cdot \sum \rho_1 \cdot \mu_1}{\sum \rho_1 \cdot \sum \rho_2} - \frac{\sum \rho_1 \cdot \sum \rho_2 \cdot \mu_2}{\sum \rho_1 \cdot \sum \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum \rho_2 \cdot \sum \rho_1 \cdot \mu_1 + \sum \rho_1 \cdot \sum \rho_2 \cdot \mu_2}{\sum \rho_1 \cdot \sum \rho_2}\right) \\
&= \left(\frac{\rho_1 \cdot \rho_2}{\sum \rho_1 \cdot \sum \rho_2}\ ,\ \mu_1 + \mu_2 - \frac{\sum\sum \rho_1 \cdot \rho_2 \cdot \mu_1 + \sum\sum \rho_1 \cdot \rho_2 \cdot \mu_2}{\sum \rho_1 \cdot \sum \rho_2}\right)
\end{aligned}$$

$$\therefore LHS = RHS$$

□

Appendix B

Summing of independent random variables

Proof that summing independent random variables result in convolution.

Let X and Y be two independent random variables and $Z = X + Y$. We want to determine the probability that $Z = z$. Suppose $X = k$, then $Z = z$ if and only if $Y = z - k$. Thus $P(Z = z) = P((X = k) \cap (Y = z - k))$. Since k can be any value, it is necessary to sum (or integrate) over all the values of k; Also X and Y is independent, thus $P(X \cap Y) = P(X) \cdot P(Y)$. Therefore:

$$\begin{aligned} P(Z = z) &= \int_{-\infty}^{\infty} P(X = k) \cdot P(Y = z - k) dk \\ \therefore P(Z) &= P(X) * P(Y) \end{aligned}$$

Appendix C

Additional Simulation Parameters

Additional simulation parameters used in the Round Robin and CSMA/CA MAC protocol experiments is listed in Table C.1 and Table C.2 respectively.

Parameter	**Value**
Channel bit rate	1.375 Mbit/s
PHY header	64 bits
Payload	800 bits
Maximum number of packets a slave node is allowed to transmit per poll received	16
Simulation length	30s

Table C.1: Additional simulation parameters for the Round Robin experiments

Parameter	**Value**
Channel bit rate	11 Mbit/s
PHY header	64 bits
MAC header	96 bits
RTS/CTS	Yes
RTS payload	192 bits
CTS payload	128 bits
ACK payload	112 bits
Payload	8184 bits
SIFS	10μs
DIFS	50μs
Slot time	20μs
Simulation length	30s

Table C.2: Additional simulation parameters for the CSMA/CA experiments

List of References

[1] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 2009.

[2] D. Barber, *Bayesian Reasoning and Machine Learning*, 2012.

[3] R. Cowell, P. Dawid, S. Lauritzen, and D. Spiegelhalter, *Probabilistic Networks and Expert Systems*, 1999, ch. 8.

[4] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 2009, ch. 22.

[5] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 2010.

[6] W. Stallings, *Data and Computer Communications*, 2004.

[7] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 2012.

[8] X. Wang and S. A. Mujtaba, "Performance Enhancement of 802.11 Wireless LAN for Asymmetric Traffic Using an Adaptive MAC Layer Protocol," in *Proceedings of Vehicular Technology Conference.* IEEE, Sep. 2002, pp. 753--757.

[9] G. Bianchi, L. F. Lawall, and M. Olivieri, "Performance Evaluation and Enhancement of the CSMA/CA MAC protocol for 802.11 Wireless LANs," in *Proceedings of Personal, Indoor and Mobile Radio Communications.* IEEE, Oct. 1996, pp. 392--396.

[10] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 2009, ch. 23.

[11] D. Barber, *Bayesian Reasoning and Machine Learning*, 2012, ch. 7.