TRANSPORT LAYER RATE CONTROL PROTOCOLS FOR WIRELESS SENSOR NETWORKS: FROM THEORY TO PRACTICE

by

Avinash Sridharan

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2009

Copyright 2010 Avinash Sridharan
Dedication

To my mother, whose wonderful memories always carry me through every adversity.
Acknowledgements

This thesis is not a singular effort. Quite a few chapters are the result of collaborative efforts with my advisor Prof. Bhaskar Krishnamachari, and my colleague/lab-mate Scott Moeller. Specifically, parts of Chapter 4, and the entire Chapter 5, were joint efforts with Bhaskar. A part of Chapter 4 was first presented at the “Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless” (WiOpt), in Seoul, in 2009 [79]. Chapter 5 has been accepted for publication in “ACM Sensys, UC Berkeley, California, USA, 2009” [77]. Chapters 6 and 7 were joint efforts with both Scott and Bhaskar. Chapter 6 was first presented at WiOpt, in Berlin, in 2008 [78], and Chapter 7 was first presented at the “Information Theory and Applications Workshop” (ITA), in San Diego, in 2009 [80,81].

Apart from the visible contributions cited above, this thesis would not have been completed without the qualitative aspects of Bhaskar’s efforts; his guidance, encouragement and support were key to the successful completion of this thesis. I have to thank Bhaskar for instilling a belief in my own abilities, for encouraging me to strive for the best, for sharing his passion for research, and for teaching me to show care and respect to my peers and colleagues. During my endeavors to achieve this goal, I have fallen more than once and every time he has stood by me, patiently supporting me, helping me regroup myself, and guiding me towards the finish line. Along with Bhaskar, another set of people who have played a critical role in the completion of this thesis
are my father and my brother Ashwin. Their understanding and unflinching support gave me the bandwidth to pursue and finish this task.

I have to thank all my friends who provided me with an environment of hope, and a sense of belief that made the hardest of tasks seem all the more enjoyable, all the more worthwhile to accomplish. Ashlesha and Neeraj, who gave me a home away from home. Their passion for their work, and life in general, always made me strive for something better; both in terms of the effort towards my thesis, and in my relationship with my friends and colleagues. Sushila, one of my dearest friends, who always bought out a childish exuberance in me; keeping my inquisitive nature intact, making me see the bright side of every situation, and nudging me along whenever I felt like hitting a road block. Siddharth and Anuraag, my friends from the days I pursued my Masters at USC, whose homes always acted as a wonderful get-away, and helped me re-energize for pursuing this goal. Atul and Aman, my friends from my undergrad, who never let me loose my connection with the past, and who kept reminding me of all the wonderful times in the past and present; making me hopeful towards a better future. Nupur and Vivek, my housemates, who made life at USC all the more memorable.

I have to thank all the group members of the Autonomous Networks Research Group, past and present, who helped me grow as a person, made me feel part of something special. Specifically Marco and Shyam, for mentoring me and looking out for me, and Scott, for the numerous discussions and collaborations that made the experience all the more richer.

I have to thank the faculty at USC, for the wonderful set of courses they taught me; which helped me create a strong foundation for completing this thesis. Especially Prof. Michael Neely and Prof. Ramesh Govindan who, apart from their courses, gave me invaluable advice/comments greatly helping me improve the quality of this thesis.
Last but not the least, I have to thank my mother, Vijayalaskhmi Sridharan, of whom all that I
have left are some wonderful memories, but without whom I would never have become the person
I am today.
# Table of Contents

Dedication ii  
Acknowledgements iii  
List Of Tables ix  
List Of Figures x  
Abstract xiii  

## Chapter 1: Introduction  
1.1 Wireless Sensor Network (WSN) 2  
1.2 Capacity region of a network 4  
1.3 The need for a rate control protocol 6  
1.4 Understanding the need for cross-layer design in wireless networks 8  
1.5 Cross-layer design as an optimization problem 10  
1.6 Understanding requirements for transport layer rate control in WSN 12  
1.7 Contributions 13  
1.8 Organization 15

## Chapter 2: Defining Objectives for a Rate Control Protocol  
2.1 Maximizing network utilization 17  
2.2 Lexicographic max-min fairness 18  
2.3 Proportional fairness 19  
2.4 Summary 20

## Chapter 3: State of the Art  
3.1 Congestion control in wired networks 22  
3.2 Congestion control in 802.11-based multi-hop wireless networks 25  
3.3 Congestion control in wireless sensor networks 28  
3.4 Using cross layer optimization for rate control design 30  
3.5 Backpressure-based rate control mechanisms 32  
3.6 Capacity models for wireless networks 32  
3.7 Positioning of our contributions 34
### Chapter 4: Receiver Capacity Model

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Modeling the capacity region for a Wireless Sensor Network (WSN)</td>
<td>38</td>
</tr>
<tr>
<td>4.2 The Receiver Capacity Model (RCM)</td>
<td>43</td>
</tr>
<tr>
<td>4.3 Feasibility of RCM</td>
<td>46</td>
</tr>
<tr>
<td>4.3.1 Receiver Capacity Model vs Clique Capacity Model</td>
<td>49</td>
</tr>
<tr>
<td>4.3.1.1 Failings of CCM</td>
<td>50</td>
</tr>
<tr>
<td>4.3.1.2 Failings of RCM</td>
<td>51</td>
</tr>
<tr>
<td>4.3.2 Assumptions and definitions</td>
<td>53</td>
</tr>
<tr>
<td>4.3.3 The sufficiency condition</td>
<td>55</td>
</tr>
<tr>
<td>4.4 Justifying the use of the receiver capacity model</td>
<td>58</td>
</tr>
</tbody>
</table>

### Chapter 5: Designing an Explicit and Precise Rate Control Protocol

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Software architecture for WRCP in TinyOS-2.x</td>
<td>64</td>
</tr>
<tr>
<td>5.2 The Wireless Rate Control Protocol (WRCP)</td>
<td>65</td>
</tr>
<tr>
<td>5.2.1 The WRCP algorithm</td>
<td>66</td>
</tr>
<tr>
<td>5.2.2 Rate update interval</td>
<td>70</td>
</tr>
<tr>
<td>5.2.3 Estimating receiver capacity</td>
<td>71</td>
</tr>
<tr>
<td>5.2.4 Estimating neighborhood active flow counts</td>
<td>71</td>
</tr>
<tr>
<td>5.2.5 Estimating transmission rates</td>
<td>72</td>
</tr>
<tr>
<td>5.2.6 Estimating sender link quality</td>
<td>73</td>
</tr>
<tr>
<td>5.2.7 Estimating external interference</td>
<td>73</td>
</tr>
<tr>
<td>5.2.8 Rate bootstrapping for flow joins</td>
<td>74</td>
</tr>
<tr>
<td>5.3 Parameter selection</td>
<td>75</td>
</tr>
<tr>
<td>5.4 Experimental setup</td>
<td>78</td>
</tr>
<tr>
<td>5.5 Stability analysis</td>
<td>79</td>
</tr>
<tr>
<td>5.6 Comparative evaluation</td>
<td>82</td>
</tr>
<tr>
<td>5.6.1 IFRC</td>
<td>82</td>
</tr>
<tr>
<td>5.6.2 Evaluation metrics and hypotheses</td>
<td>85</td>
</tr>
<tr>
<td>5.6.3 Comparative evaluation methodology</td>
<td>85</td>
</tr>
<tr>
<td>5.6.4 Static scenario</td>
<td>86</td>
</tr>
<tr>
<td>5.6.5 Dynamic scenario</td>
<td>87</td>
</tr>
<tr>
<td>5.6.5.1 Case 1</td>
<td>90</td>
</tr>
<tr>
<td>5.6.5.2 Case 2</td>
<td>92</td>
</tr>
<tr>
<td>5.6.6 WRCP performance in the presence of external interference</td>
<td>95</td>
</tr>
<tr>
<td>5.7 Summary and Discussion</td>
<td>98</td>
</tr>
</tbody>
</table>

### Chapter 6: Building a Flexible Rate Control Stack

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Lyapunov optimization formulation</td>
<td>102</td>
</tr>
<tr>
<td>6.1.1 Lyapunov optimization with receiver capacity virtual queues</td>
<td>104</td>
</tr>
<tr>
<td>6.1.1.1 Transmission Control decision</td>
<td>109</td>
</tr>
<tr>
<td>6.1.1.2 Admission control decision</td>
<td>109</td>
</tr>
<tr>
<td>6.2 Implementing Backpressure-based Rate Control using virtual queues</td>
<td>110</td>
</tr>
<tr>
<td>6.2.1 Hardware/Software implementation details</td>
<td>112</td>
</tr>
<tr>
<td>6.2.2 An illustrative example</td>
<td>115</td>
</tr>
<tr>
<td>6.3 Experimental evaluation</td>
<td>117</td>
</tr>
</tbody>
</table>
# List Of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Variables used in the WRCP algorithm. formulation.</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>The change in theoretical bound of $\alpha$, with change in topology.</td>
<td>80</td>
</tr>
<tr>
<td>6.1</td>
<td>Variables used in the Lyapunov formulation.</td>
<td>104</td>
</tr>
<tr>
<td>6.2</td>
<td>Weights and topologies for each of the three test scenarios.</td>
<td>118</td>
</tr>
<tr>
<td>6.3</td>
<td>Common parameters for all test scenarios.</td>
<td>118</td>
</tr>
<tr>
<td>6.4</td>
<td>Optimal packet rate allocations subject to receiver bandwidth</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>constraints for scenarios 0 through 2.</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Achieved packet rate allocations for experimental scenarios 0 through</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>The sum-log utilities achieved for the multi-hop topology using the</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>proportional fair controller with the PDQ and the MDQ schemes.</td>
<td></td>
</tr>
</tbody>
</table>
List Of Figures

1.1 A typical sensor network device. . . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.2 A collection tree. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.3 A 1-hop topology. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.4 The rate region of the 1-hop topology. . . . . . . . . . . . . . . . . . . . . . . 6
4.1 The capacity region of two flows over a single wired link. . . . . . . . . . . . . 40
4.2 A single-hop wireless topology. . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
4.3 The capacity region of two flows in a broadcast domain of an ALOHA network. . 42
4.4 Saturation throughput for multiple senders for the TinyOS CSMA MAC. . . . . . . 44
4.5 An illustrative example of the receiver capacity model . . . . . . . . . . . . . . . 46
4.6 The pentagon topology. This topology helps explains the limitations of CCM. . 50
4.7 The assymetric topology. This topology helps explains the limitations of RCM. . 52
5.1 Rate allocation and queue backlog behavior for IFRC . . . . . . . . . . . . . . . 62
5.2 The behavior of allocated rate and queue backlog for WRCP. . . . . . . . . . . . 63
5.3 Software Architecture for WRCP . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
5.4 A single time step execution of the WRCP protocol at node i. . . . . . . . . . . . 69
5.5 A 20-node topology. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
5.6 A 40-node topology. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
6.8  Goodput for MDQ scheme using different flow controllers.  . . . . . . . . . . . 127
6.9  Goodput for PDQ scheme using different flow controllers.  . . . . . . . . . . . 129
7.1  The software architecture of a backpressure based rate control stack.  . . . . . 134
7.2  Understanding backpressure scheduling.  . . . . . . . . . . . . . . . . . . . . 136
7.3  The 20-node routing tree  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143
7.4  The 40-node routing tree  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144
7.5  Connectivity for 20 and 40-node topologies.  . . . . . . . . . . . . . . . . . . . 145
7.6  Selection of the parameter V for different topologies.  . . . . . . . . . . . . . 146
7.7  Log utility performance of MAC protocols across different topologies.  . . . . 147
7.8  Average total queue length for MAC protocols across different topologies.  . . 148
7.9  Goodput of IFRC and backpressure-based stack in a static scenario.  . . . . . 150
7.10 Queue-length of IFRC and backpressure-based stack in a static scenario  . . . . 152
7.11 Behavior of PDQ and IFRC under dynamic flow scenario.  . . . . . . . . . . . 154
7.12 Goodput of IFRC, PDQ, and MDQ MAC under dynamic flow scenario.  . . . . 155
7.13 Behavior of PDQ under dynamic flow scenario.  . . . . . . . . . . . . . . . . . . 157
7.14 Goodput performance of IFRC, PDQ and MDQ MAC with adaptive V.  . . . . 158
Abstract

Because of limited bandwidth availability and their typical dense, multi-hop deployment, wireless sensor networks have a fundamental need for efficient transport layer rate control. State of the art transport layer rate control protocols in wireless sensor networks are primarily heuristics that rely on Additive Increase Multiplicative Decrease (AIMD) mechanisms. This heuristic-based design of state of the art rate control protocols raises two key issues: first, due to the AIMD based mechanism the protocols suffer from long convergence times and large end-to-end packet delays; second, existing protocols are monolithic in design, either focusing purely on congestion control functionality without regard for rate utility optimization, or trying to optimize for a specific rate utility function. We improve upon the state of the art by proposing two rate control protocols that address the above issues. To address the issue of long convergence times and large end-to-end packet delays, we propose the Wireless Rate Control Protocol (WRCP). To address the issue of monolithic protocol design, we propose the Backpressure-based Rate Control Protocol (BRCP).

WRCP, to our knowledge, is the first explicit and precise rate control protocol for wireless sensor networks. WRCP has been designed using a novel interference model, called the receiver capacity model. The model helps determine the exact available capacity at each receiver in the network. WRCP uses the available capacity information presented by the model, and distributes this capacity amongst contending flows in a neighborhood in order to achieve a lexicographic
max-min fair rate allocation. The use of explicit capacity information allows WRCP to exhibit fast convergence time. The explicit capacity information also allows WRCP to operate within the capacity region, resulting in small end-to-end delays.

BRCP is a, novel, flexible rate control protocol that has the ability to optimize for any concave rate-utility function. The design of BRCP is achieved by applying Lyapunov drift based stochastic optimization techniques to a Carrier Sense Medium Access (CSMA) based MAC. The ability of BRCP to make rate control decisions purely on local queue information makes it extremely useful in a wireless sensor network, since it requires minimal control information exchange.
Chapter 1

Introduction

The aim of this thesis is to design and implement transport layer rate control functionality for a specific type of multi-hop wireless network, the Wireless Sensor Network (WSN). The objective of proposing these rate control protocols is two-fold: first, to improve upon the rate-performance of existing state of the art for rate control protocols in WSN; second, is to introduce an application-aware rate control protocol, that has the ability to present rate allocations catering to the specific requirements of an application, breaking away from the norm of application agnostic rate control design. Before presenting details of our contributions, and the problems that we address, we introduce a few key concepts that will help understand the problems related to transport layer rate control that this work aims to solve, as well as the quantitative approach used to design and implement the proposed solutions. We start by describing the type of networks that the term wireless sensor network represents, and the objective of designing and deploying these networks.
1.1 Wireless Sensor Network (WSN)

Wireless sensor networks are multi-hop wireless networks, composed of low-rate, low-power, embedded devices having sensing capabilities. These networks are targeted towards numerous types of sensing applications. A few examples of the targeted applications are: structural health monitoring [62], habitat monitoring [85], sensing seismic activity [99], health information systems [28], traffic monitoring [3], forest fire detection [104], volcanic activity monitoring [100] and many defense related applications [30]. The motivation and vision behind the use of these networks is that the low cost, small form factor, devices will allow for a larger number of sensors to be deployed, increasing the granularity of the sensing information collected. Further, the wireless nature of these networks will make deployments easier, and allow for deployments of these networks in areas where it might not be feasible to have wired infrastructure, e.g., remote isolated areas where environment monitoring needs to be carried out.

Figure 1.1(a) shows an example of a typical sensor network device, the Tmote Sky [55]. Figure 1.1(b), gives a sense of the form factor of the current devices. Figure 1.1(b) shows that
sensor networking devices have already become smaller than the more popular embedded platforms such as cell phones, and the hope is the form factor will be reduced further to the size of a penny, or may be even smaller. The form factor of these devices imposes severe constraints on their hardware capabilities. For example, the Tmote Sky platform, which runs the MSP430 microprocessor, which is a 32Khz processor, has 10K RAM and 1MB of flash.

Another important feature of wireless sensor networks, apart from limited hardware/software capabilities of the devices, is that these networks are envisioned to have an operational life span ranging from a few months to a few years on AA batteries. Thus, given the energy constraints, the devices in these networks are fitted with low-power radios. The radios typically follow the IEEE 802.15.4 standard [9], which has become the de-facto standard for wireless sensor network, and designed to be used with short range devices [9]. The low-power radios, in turn, result in small packet sizes, typically 40-100 bytes; the goodputs in these networks are in the range of 40-80 bytes/sec for network sizes of 20-40 nodes, with a network diameter of 5-7 hops\(^1\). Thus, unlike IEEE 802.11 [16] based multi-hop wireless networks wireless sensor networks are low-power, low-rate networks; having severe restrictions on the complexity in terms of the software, and hence the protocols that can be run over these networks.

A typical routing topology that is found in sensor networks is the collection tree. Figure 1.2 gives an example of a collection tree\(^2\) that might be found in a real sensor network. In Figure 1.2, node 1 is the sink. All other nodes in this network are sources, that want to send their sensed data over the multi-hop collection tree to the sink. The labels on the links represent the success probability, for a single packet transmission over a wireless link that exists between a given node

\(^1\)These numbers were obtained from experiments run on the USC Tutornet testbed [46], a 94 node wireless sensor network testbed.

\(^2\)This routing topology was obtained on the USC Tutornet testbed.
pair. The success probability provides a measure of the quality of the wireless link. During the course of this thesis, the collection tree will be the recurring routing topology in our protocol design and evaluation.

### 1.2 Capacity region of a network

A key concept necessary for understanding the role of rate control protocols in a packet switched data network (wired or wireless), is the concept of a capacity region [29] of a network. The capacity region of a network is defined as the set of rate vectors that are sustainable by the network. A rate vector is a set of rates, with each element of the rate vector corresponding to a rate allocated to a given source in the network. A rate vector $\overrightarrow{r}$ is said to be sustainable by the network if the time average queue size of every router in the network remains bounded when sources in the network are sending data to their respective destinations at the allocated rate $\overrightarrow{r}$. A source-destination pair is referred to as a flow, and hence the capacity region of a network represents the end-to-end flow rates sustainable by the network. It is important to note that the dimensionality
of the capacity region depends on the number of flows that exist in the network. Further, the boundary of the capacity region is determined by the maximum sustainable flow rates, which in turn are determined by a combination of protocols implemented at the routing and the MAC layer, and the link rates presented by the physical layer. The definition of a sustainable rate presented above is under the assumption that routers in the network have infinite queue sizes, and hence is useful only from a theoretical perspective. In practical systems, with finite queue sizes, when the network is operating at a sustainable rate the queue occupancy in the routers will be low, while at an unsustainable rate the time average value of congested queues in the network will become equal to the maximum queue size, resulting in excessive packet drops.

We illustrate the concept of a capacity region by taking the example of a simple one-hop topology, shown in Figure 1.3, with two sources. For this topology there are two sources sending data to a single destination. The capacity region of this two source network can be represented by Figure 1.4. The capacity region for this two source network is assumed to be convex [7]. The region marked feasible represents the set of sustainable rates for this network, and the region
marked infeasible represents the unsustainable rate for this network. For this particular example, since there exist only two sources, the capacity region for the network can be represented by a two-dimensional plot.

1.3 The need for a rate control protocol

If sources in a network are always operating at a feasible point in the capacity region (Figure 1.4), the queues in the network will always be bounded, and the network will operate without congestion. In a multi-hop network, however, the sources in the network are unaware of the boundary of the capacity region, and hence do not have any information about the sustainable flow-rates in the network. Thus, in a multi-hop network, it is possible that sources start trying to send data to their destinations at rates that are not sustainable by the network forcing the network to operate in the infeasible region, resulting in congestion. The goal of the rate control protocol is to try and avoid/mitigate congestion in the network by performing two key functionalities: first, it detects, and informs sources, that the network is operating in the infeasible region; second it reduces the
flow-rate of sources, bringing the network from an infeasible point of operation to a feasible point of operation, relieving congestion in the network.

The performance of a rate control protocol is measured in terms of the rate-utility presented to each source in the network. The rate-utility presented to a source is usually given by a concave function [7] of the rate allocated to the source. If the objective of the rate control protocol is purely to relieve congestion in the network, the rate control protocol can choose to operate the network at any point in the feasible rate region. However, apart from congestion control/avoidance, rate control protocols are designed to guide the network to operate at a specific point in the rate region in order to maximize the sum of the rate-utility of all sources in the network. We will present further details about the different rate-utility functions that the rate control protocol can optimize for in Chapter 2.

For low-power wireless sensor networks, the degradation of per-source sustainable rate is quite drastic with increase in network size, because of their limited bandwidths, high density deployments, and multi-hop-operation. Given the drastic reduction in network capacity with increase in the size of the network, the motivation for the need for transport layer rate control has been cited by a few deployed sensor network applications [3, 62]. We can further motivate this requirement, with the following empirical evidence. In a wireless sensor network testbed [46], it has been observed that with a 40 byte packet, a 4-node network can give per-source rate as low as 16 pkts/sec. A 20-node network under similar conditions results in a reduction of per-source rate to ∼ 2 pkts/sec, and in a 40-node network this rate reduces to ∼ 0.5 pkts/sec. Thus, even if low data rate sources are operational in these network (which is typical for a WSN), given the scarce capacity, a sufficiently large number of such sources can easily cause congestion collapse.
if they are operating unaware of the sustainable rates in the network. This observation reflects the importance of rate control protocols in these networks.

1.4 Understanding the need for cross-layer design in wireless networks

Every device capable of communicating over a network has a communication module implemented in software or hardware (or a mix of both) that is referred to as the network stack. The network stack consists of components that provide functionalities (not necessarily all of them), such as the transport layer functionality, the routing layer, the MAC and the physical layer, in order to allow the device to communicate over the network. The design of network stacks, targeted towards current networking infrastructure for packet switched data networks, has primarily been based on the Open Systems Interconnection (OSI) model [64]. A good example of a network stack that is based on the design philosophy of the OSI model is the TCP/IP stack [64]. Under this model the functionality of the network stack is segregated into different layers; each layer is fully self-contained, such that in order to perform its functionality a particular layer does not rely on any other control information, except for the minimalist control information for acknowledging a packet injected into the underlying layer. The control information for acknowledging a packet is also binary in nature, only informing the upper layer whether a packet has been received or not, without explicitly stating the reason for the loss of the packet. To list a few reasons for packet losses, e.g., packet losses can occur due to network level congestion, MAC level collisions, or bit errors during transmission at the physical layer.

While the OSI-based architecture of current network stacks (TCP/IP) has served well in the context of wired networks—in terms of the performance presented by the network stack, as well
as the operational flexibility of the stack [31] (this observation is captured by the well known adage “everything runs over IP, and IP runs over everything”)—the gains presented by this architecture diminish significantly when the same network stack is run over multi-hop wireless networks. This observation is particularly true when considering the specific case of transport layer rate control for multi-hop wireless networks. The performance degradation of transport layer rate control in an OSI-based stack such as TCP/IP, over multi-hop wireless networks, is attributed to the minimalist control information exchange between layers of the network stack [2].

In wired networks the binary feedback, in terms of packet acknowledgement control information, works well since network topology in wired networks are inherently stable, and the underlying link reliability is exceptionally high; making congestion the primary cause for loss of packets in wired networks. For wired networks, OSI-based transport layer rate control protocols such as TCP, thus, interpret the feedback of packet loss correctly as a signal of congestion, and force sources to cut back their rates performing congestion control functionality efficiently in these networks. In wireless networks, however, the frequent change in network structure (due to mobility or link reliability), and the higher link error rates (as compared to wired networks), makes the binary packet acknowledgement control information exchange between layers inadequate. In a wireless network, apart from congestion, a packet could have been lost due to link unreliability, or due to collisions that might be caused by interference. Thus, due to lack of information, OSI-based transport layer rate control protocols such as TCP force sources over a wireless network to woefully underperform, in terms of the flow rates that the sources can achieve [2, 18].

Given the failings, due to the OSI-based design, of current network stacks for multi-hop wireless networks, there has been a shift towards a design philosophy that promotes a more fine
grained information exchange between layers. This is referred to as the cross-layer design philosophy. In this approach, the transport, routing, and MAC layer are no longer segregated entities\(^3\), but rely on explicit and fine grained information exchange between the layers to carry out the functionality assigned to each layer. The cross-layer design philosophy has strong theoretical foundations, promoted specifically by the seminal works by Kelly et al. [43], and Low and Lap-sley [51]. Despite the progress made in developing theoretical frameworks that will enable the design of cross-layer stacks in multi-hop wireless networks, when considering transport layer rate control design, to the best of our knowledge, there are no real world implementations that employ this design philosophy. The aim of this thesis is to design and implement a cross-layer stack with emphasis on enabling transport layer rate control functionality for a specific type of multi-hop wireless network, the wireless sensor network. We will now introduce the theoretical foundations that enable a quantitative approach to cross-layer design of transport layer rate control for wireless sensor network.

1.5 Cross-layer design as an optimization problem

Since the focus of this thesis is on transport layer rate control, we concentrate on presenting a quantitative description of the approach taken to enable cross-layer design of the rate control protocol. A key functionality of the rate control protocol at the transport layer is to inform sources about the sustainable rate at which they can inject data into the network. The strategy followed by the rate control protocol for achieving a specific rate allocation is defined by the rate-utility objective (Section 1.3) the protocol is aiming to optimize. The rate-utility objectives targeted by

\(^3\)In this thesis we assume the physical layer is a fixed entity, and do not consider cross-layer optimization approaches that also vary parameters at the physical layer, though these have been explored before in the wireless literature [36, 70, 73].
the rate control protocol can be to maximize the efficiency of the network, to guarantee a fair rate allocation to all sources in the network, or to provide a balance between efficiency and fairness in terms of allocated rates. As mentioned in Section 1.3, the rate-utility for a source $i$ is usually a concave function of the allocated rate $r_i$ given by $g_i(r_i)$. The exact form of the function $g_i(r_i)$ is determined by the objective that the rate control protocol is striving to achieve. The cross-layer design of the transport layer rate control can then be posed as a constrained optimization problem in the following form:

$$
\text{P} : \max \sum_{r_i \in \mathbb{R}} g_i(r_i)
$$

$$
\text{s.t. } \mathbb{R} \in \Lambda
$$

Here $\Lambda$ is the capacity region of the network. If the objective in the above optimization problem is concave, and the capacity region $\Lambda$ is convex, then the optimization problem is convex [7] and we can find a unique rate vector $\mathbb{R}$, within the capacity region, that can maximize the global objective function $\sum_{r_i \in \mathbb{R}} g_i(r_i)$.

If for a given network we are able to formulate the rate control problem in the above form, we can use a centralized convex programming solver to obtain solutions to this problem, thus performing rate control in a centralized manner. However, in general, since centralized solutions are not flexible to network dynamics (such as mobility, flow dynamics, node failures, link failures, etc.), and do not scale well with the size of the network$^4$ [15], we focus on designing rate control algorithms that can achieve a solution to the above optimization problem in a distributed manner.

$^4$While it’s a broadly accepted notion that decentralized protocol design is a more efficient design principle, there are instances when centralized design principles have been promoted. Examples are solutions to the coverage problem in wireless sensor networks [52], enterprise management of campus wide networks [10], and even in rate control for heterogeneous wireless sensor networks [63, 71].
Distributed algorithms have the ability to make local decisions based on neighborhood information, such that these local decisions will result in a global optimum, making the algorithms adaptable to network dynamics, and scalable with the size of the network.

From a theoretical perspective, the seminal works by Kelly et al. [43], and Low and Lapsley [51], highlighted the power of using duality-based techniques in finding cross-layer distributed solutions to the problem $P$, at least in a wired setting. These ideas were then used to formulate theoretical proposals for such solutions in a wireless setting as well ([13], [95]). More recently, the works by Stolyar [83] and Neely et al. [58] have built upon these seminal contributions, and have shown that elegant, and modular distributed solutions to the problem $P$ can be designed by appropriately modifying the MAC layer, and allowing sources to make rate allocation decisions purely based on the occupancy of their local queues. Despite the advances made in theoretical understanding of the problem $P$, it is only recently that attempts are being made to convert these theoretical contributions into real world protocols, especially in the context of multi-hop wireless networks.

1.6 Understanding requirements for transport layer rate control in WSN

Given the need for congestion control, there have been numerous proposals ([93], [67], [63], [101], [71]) that can perform congestion control in these networks. However, apart from the core functionality of congestion control, we believe rate control protocols targeted towards a WSN should exhibit two important characteristics. First, they should be responsive, exhibiting the ability to inform sources about the sustainable rates in as small a duration as possible. The
need for rate control protocols to be responsive is primarily due to energy efficiency concerns. Since in these networks, apart from bandwidth, energy is also a scarce resource, and radios are considered to be the primary consumers of energy, it is imperative to reduce the active duration of these radios. This can be achieved by ensuring short flow completion times, since short flows directly imply shorter uptime for the radios. If the rate control protocols are responsive, they will be able to inform sources about the sustainable rates in a shorter duration, allowing the sources to transmit at a faster rate in a much shorter time frame leading to shorter flow completion times.

Second, rate control protocols for WSN should have the flexibility to optimize for different types of rate-utility functions. This characteristic is important, particularly in a sensor network setting, due to the diverse nature of the applications envisioned to be operated over these networks. Thus, rate control protocols in WSN should not be designed to optimize for a single utility function, rather the rate control stack should be flexible enough, having the ability to optimize for any utility function presented by the application.

State of the art designs for rate control protocol in wireless sensor networks are however primarily heuristics, that rely on additive increase multiplicative decrease (AIMD) mechanisms. They suffer from long convergence times and large end-to-end packet delays. In addition to being based on heuristics, state of the art rate control protocols for wireless sensor networks are monolithic in design. The monolithic protocol design caters for optimizing a specific utility function (such as max-min fairness), or performing purely congestion control functionality without regard for optimization. Thus, although state of the art rate control protocols for WSN perform the core task of congestion control, they are deficient in terms of the above mentioned desired characteristics for such protocols in a WSN context.
1.7 Contributions

The contributions of this thesis are two rate control protocols that aim to address the deficiencies of state of the art for rate control in WSN, as discussed in Section 1.6. The novelty in both the protocols is the quantitative approach we have pursued in our design, allowing us to realize the powerful theoretical frameworks proposed for achieving a distributed solution to the problem $P$ (Section 1.5) in practice. The first, called the **Wireless Rate Control Protocol (WRCP)**, addresses the issue of convergence times and end-to-end packet delay using explicit capacity information. The Wireless Rate Control Protocol presents a distributed solution to the lexicographic max-min fairness problem over a collection tree. It uses a novel interference model, called the receiver capacity model, in order to define the capacity region of a wireless sensor network employing a Carrier Sense Medium Access with Collision Avoidance (CSMA-CA) MAC. The use of this model allows WRCP to achieve fast convergence time and small end-to-end packet delays.

The second, called the **Backpressure-based Rate Control protocol (BRCP)**, presents a flexible rate control protocol that can optimize for any global concave objective function using purely local queue information. The backpressure-based rate control protocol presents a more generic solution to the problem $P$. It has been designed using stochastic optimization techniques that allow it to optimize for any concave utility function ($g(r_i)$) by simply using local queue information. The generic solution provides different, rate-utility specific, transport layer, **flow controllers** to work with this protocol. Each flow controller aims to optimize a specific application-defined concave rate-utility function. These different flow controllers can be presented to the application developer as a set of libraries, allowing the user the flexibility to pick and choose the flow controller, to be used with BRCP, that will aim to optimize his application specific rate-utility.
function. Both, WRCP and BRCP, highlight the gains that can be achieved by pursuing a more theoretical approach to rate control design for wireless networks in general, and wireless sensor networks specifically.

1.8 Organization

The organization of this thesis is as follows: In Chapter 2 we proceed to explain quantitatively the different objectives that a rate control protocol can strive to achieve. In Chapter 3, in order to put the contributions of thesis into perspective, we present prior art pertaining to rate control design in wired and wireless networks. In Chapter 4 we present a novel interference model called the receiver capacity model that will allow us to define the constraints of the problem P(Section 1.5), and hence will form the crux of the analytical framework on which the protocols proposed in chapters 5 and 6 are based. In Chapter 5 we present the Wireless Rate Control Protocol (WRCP) protocol, an explicit and precise rate control protocol, that has the ability to achieve a lexicographic max-min fair rate allocation over a WSN collection tree. In Chapter 6 we present the design and implementation of the Backpressure-based Rate Control Protocol (BRCP). BRCP relies on a collection of flow controllers that can be presented to the user as a library, each flow controller having been designed to optimize for an application specific concave rate-utility function. In Chapter 7, we present a rigorous empirical evaluation of this backpressure-based protocol (BRCP), to understand the various design choices that can be made, in terms of the algorithms and parameters, while implementing the MAC and Transport layers of this stack in a WSN setting. Finally, in Chapter 8, we present a summary of the contributions of this work, as
well as a discussion to highlight the various open problems associated with this work and which can form the foundation of future research based on this thesis.
Chapter 2

Defining Objectives for a Rate Control Protocol

In Chapter 1, we showed that a cross-layer design of a transport layer rate control protocol can be undertaken, by formulating the problem in the following form:

\[ P_1 : \max \sum_{r_i \in \mathcal{R}} g_i(r_i) \]

s.t. \( \mathcal{R} \in \Lambda \)

Assuming that the constraints are well defined (in Chapter 4, we show how we quantify these constraints for a wireless sensor network), in order to use the formulation \( P1 \) to design rate control protocols for a network, we need to define the objective function \( g_i(r_i) \).

In general the rate-utility function \( g_i(r_i) \) can be any concave function. Thus, there is an infinity of objectives that can be defined for rate control protocols. However, in this thesis we shall focus on three specific objectives. We aim to design a rate control protocol that can either try to maximize network utilization, or it can try to optimize for fairness, or it can try to find a good trade-off between network utilization\(^1\) and fairness, by trying to maximize utilization while achieving a certain level of fairness. We focus our efforts on these three objectives, since it

\(^1\)We will be using the terms network utilization and network utilization interchangeably.
has been shown that, in general, the objectives of maximizing network utilization and achieving fairness are diametrically opposite [66,88], and hence represent the two extremes of the spectrum of possible objectives. The third objective, of achieving a balance between maximizing utilization and fairness, finds a middle ground between these two extremes. Thus, we believe these three objectives are a good sample, from the entire spectrum of objectives, to showcase the cross-layer design of transport layer rate control protocols that aim to solve the problem P1.

We now discuss the three objective functions that a rate control protocol can use in order to maximize network utilization, to optimize fairness, or to achieve a good balance between fairness and utilization.

### 2.1 Maximizing network utilization

In order to maximize for network utilization the rate control problem can defined as:

\[ P2 : \max_{\mathbf{r} \in \mathbf{r}} \sum_{i \in \mathbf{r}} r_i \]

\[ \text{s.t. } \mathbf{r} \subset \Lambda \]

As mentioned earlier, maximizing network utilization requires the maximization of sum rate. In wireless networks solutions for the above optimization problem would generally involve allocating all the rate to one or more of the first hop nodes. As is evident, such a solution is inherently unfair (since any node farther then a single hop will not be able to communicate), making solutions to this problem quite useless in practice.
2.2 Lexicographic max-min fairness

A lexicographic max-min fair rate vector $\overrightarrow{r}^*$ is a rate vector, such that if the elements of the max-min fair rate vector $\overrightarrow{r}^*$ are arranged in ascending order, for any other sustainable rate vector $\overrightarrow{\tilde{r}}$ whose elements are also arranged in ascending order, $\overrightarrow{r}^*$ will always be lexicographically greater than $\overrightarrow{\tilde{r}}$, i.e. there exists a $j$, such that $r_j^* > \tilde{r}_j$, and $\forall i, 0 < i < j, r_i^* = \tilde{r}_i$.

The physical interpretation of the above definition, for lexicographic max-min fairness, can be better understood by taking the example of max-min fairness in wired networks. In the case of wired networks, a necessary and sufficient condition for a lexicographic max-min fair rate vector is the following: a rate vector is said to be max-min fair, if and only if every flow in the network has at least one bottleneck link. A flow is said to be bottlenecked at a link, if the flow has the highest rate amongst all flows sharing this link and the capacity of the link has been exhausted. A proof for the necessary and sufficient condition, for a wired network, can be found in [4].

An intuitive understanding of lexicographic max-min fairness can be obtained by the following example: imagine a wired network where each node in the wired network is replaced by a vessel of certain capacity, and the links are replaced by pipes. If sources were to pump water into these pipes from the leaves of this network, each of the flows can keep increasing its rate till they hit a vessel whose capacity has been exceeded. The first such vessel for a given flow is called the bottleneck link for that flow. All flows that have the same bottleneck link will share the same rate.
Max-min fairness can be quantified as an optimization problem by using the notion of \( \alpha \)-fairness. \( \alpha \)-fairness is defined as follows:

\[
P_3 : \max \sum_{r_i \in \mathbf{r}} \frac{r_i^{1-\alpha}}{1-\alpha} \quad s.t \quad \mathbf{r} \in \Lambda
\]

As \( \alpha \to \infty \), it can be shown that the solution to the above optimization problem satisfies the necessary and sufficient condition of a lexicographic max-min fair vector. In Chapter 6, we show that in practice, for sensor networks, the solutions for \( \alpha \)-fairness start approaching the max-min fair solution even for \( \alpha \) values as small as 5.

### 2.3 Proportional fairness

A well known utility function that is able to achieve a good trade-off between fairness and utilization is the log rate-utility function [66]. This is also known as proportional fairness. A rate control problem solving for proportional fairness is defined as follows:

\[
P_4 : \max \sum_{r_i \in \mathbf{r}} \log(r_i) \quad s.t \quad \mathbf{r} \in \Lambda
\]

Note that since the objective function is a sum of logarithms, maximizing the objective function implies maximizing the product of the rates (\( \prod_{r_i \in \mathbf{r}} (r_i) \)). In order to maximize the product of rates, the solution has to ensure that there is no source who is starved (given a zero rate). Also, sources that create more congestion, or result in larger backlogs, should be given a smaller rate. Intuitively, sources that have a larger hop count to the destination will require to traverse more
links resulting in consumption of larger amounts of capacity (resulting in higher amounts of con-
gestion), as compared to sources that are closer to the destination. Proportional fairness thus tries
to be partial to sources that are closer to the destination, while ensuring that there is no source that
is starved. Proportionally fair solutions thus try to satisfy the inherent tension that exist between
the problems of maximizing network utilization and optimizing fairness [66].

2.4 Summary

In this chapter we quantified the objective $\sum_{r_i \in \mathcal{R}} g_i(r_i)$ that rate control protocols developed in this
thesis will strive to optimize. While there are an infinity of objectives that rate control protocols
can be designed to optimize, in this thesis we focus on three specific objectives: maximizing
network utilization, max-min fairness and proportional fairness. We focus on these three objec-
tives since the objective of maximizing network utilization and max-min fairness represent two
extremes, of the spectrum of objectives that a rate control protocol can try to optimize, while
proportional fairness represents a good trade-off between maximizing network utilization and
max-min fairness. We therefore believe that the three objectives are a good sample of the entire
spectrum of objectives.

As will be seen in the following chapters, using theoretical frameworks that solve for the
generic rate control problem $P1$ result in elegant design of rate control protocols. Despite the
elegant design, and the strong theoretical foundations for these designs, that result from for-
mulating the rate control problem as a constrained optimization problem $P1$, it is important to
understand the limitations of this mathematical formulation. A key limitation of the problem $P1$
is that the optimal allocated rate $r_i^*$, that will maximize the global concave rate-utility function
\[ \sum_{r_i \in R} g_i(r_i), \] is a long-term time-average rate. This implies that this formulation is relevant only for long flows. It is unclear what this formulation says about short flows. Another limitation of this formulation is that it does not represent the class of problems, where rate control protocols need to be optimized for delay instead of throughput. In order to solve the problem \( P_1 \) optimally, the optimal rate vector \( \vec{r}^* \) will make the constraints of the problem \( P_1 \) tight. This implies that the optimal rate vector, that will solve for the problem \( P_1 \), will lie on the boundary of the rate region. Operating at this optimal rate vector will therefore involve larger queue sizes, than operating at a sub-optimal rate vector within the rate region. This inherent tension between optimizing rate-based utility function and delay, is the reason why \( P_1 \) cannot be used to represent the class of rate control problems that want to optimize for delay.
Chapter 3

State of the Art

3.1 Congestion control in wired networks

The seminal work by Jacobson [35] was one of the first congestion avoidance mechanisms built into the Internet. The work by Jacobson [35] introduced several algorithms, such as the slow-start mechanism, and the Additive Increase Multiplicative Decrease (AIMD) algorithm that form the core of today’s Transmission Control Protocol (TCP) [64]. The version of TCP that first incorporated the algorithms proposed by Jacobson [35], is known as TCP Tahoe and made its appearance in the Internet as part of the BSD 4.3 operating system’s network stack [8]. The slow-start and AIMD mechanism, that are core to the congestion avoidance algorithm in TCP, perform end-to-end congestion control by using loss of packet acknowledgements as a sign of congestion in the network. These algorithms allow TCP to treat the network as a black box, giving it the ability to perform congestion control irrespective of the type of network.

TCP Tahoe was considered to be too conservative in its multiplicative decrease mechanism, forcing slow start whenever a packet is lost (even after the packet is successfully retransmitted),
thus reacting over-aggressively to transient congestion and hurting TCP throughput. The subsequent versions of TCP Tahoe, such as Reno and SACK [22], improve upon the performance of TCP Tahoe by introducing mechanisms that allow it to detect the transient congestion in the network better; avoiding over-throttling of TCP throughput when intermittent packet losses occur due to transient congestion. Another version of TCP, called TCP Vegas [8], was developed to improve significantly on the throughput performance of TCP Reno. TCP Vegas achieves this improvement by introducing a new congestion avoidance scheme. TCP Vegas improves the congestion avoidance mechanism by introducing a new bandwidth estimation scheme that looks at the actual throughput and estimated throughput of TCP to regulate its congestion window size, in order to better estimate the available bandwidth and avoid congestion. The key idea behind TCP Vegas is that in a congestion free network the difference between the actual and estimated throughput will be negligible, while in a congested network the difference quantifies the amount of congestion in the network, and the congestion window should be reduced by this amount. Compared to the fine grained bandwidth estimation of TCP Vegas, the multiplicative decrease mechanism used by TCP Reno to avoid congestion is too aggressive — it cuts its rate by half whenever its sees packet losses — resulting in TCP Vegas outperforming TCP Reno in terms of achieved throughput.

While the end-to-end congestion control mechanism of TCP made it network agnostic — allowing applications to run over networks, irrespective of the type of the network — it also presented some severe drawbacks. In high-speed connections with large delay-bandwidth products, the end-to-end congestion mechanism of TCP resulted in large queues, and hence high delays; since, in the absence of explicit feedback from the network TCP relies solely on packet drops for detecting congestion, resulting in queues growing to large values when TCP operates at high
throughput. Further, TCP introduced periodicity into the traffic, forcing global synchronization of the additive increase multiplicative decrease phases of traffic, leading to severe underutilization of the network. To counter these drawbacks, resulting in large delays and global synchronization of traffic, Active Queue Management (AQM) techniques were proposed. In AQM, the intermediate routers take part in congestion avoidance, by observing their queue sizes and proactively marking packets with *congestion bits*, or by proactively dropping packets to send congestion signal back to the end hosts. By introducing randomness into the marking/dropping of packets AQM techniques were able to reduce forwarding queue sizes and avoid the global synchronization problems introduced by the end-to-end congestion control mechanism of TCP. One of the first AQM mechanisms was the *Random Early Detection* (RED) algorithm proposed by Floyd *et al.* [25]. The key problem with AQM techniques such as RED and SRED [91], is that their performance is extremely sensitive to parameter settings. Subsequently proposed algorithms, such as BLUE [24], AVQ [45] and PI [33], present a more control-theoretic design of AQM mechanisms whose parameters can be estimated on the fly (based on traffic load, and traffic pattern).

An alternative to AQM, for improving the performance of end-to-end congestion control mechanism of TCP in large delay-bandwidth products was proposed in the form of FAST TCP [98]. FAST TCP is a variant of TCP, employing end-to-end congestion control, where the window control mechanism is based on a delay-based estimation technique, rather than a loss-based estimation technique as is the case with the existing implementations of TCP. The window control algorithm uses an estimate of the queueing delay experienced by the packets, to determine the number of packets that are buffered in the network; the amount by which the window is incremented or decremented depends on how many buffered packets exist in the network when
compared to a targeted buffer value. The design of FAST TCP is pursued in [98] by Jin et al., using a constrained optimization-based [7] framework. The quantitative approach taken by Jin et al. allows the authors to prove the stability and proportional fairness of the delay-based estimation technique for congestion control, used in FAST TCP, when operating in high delay-bandwidth product networks.

In the recent past there have been proposals promoting a paradigm shift in the philosophy behind congestion control in the internet; promoting the idea of moving away from end-to-end congestion, treating the network as a black box, and making congestion control more router-centric. In the Internet context, recent protocols such as XCP [41] and RCP [19] have highlighted the gains of using precise feedback using network capacity information, as compared to traditional AIMD approaches followed by TCP and its many variants. XCP [41] is a window based protocol that presents improved stability in high delay bandwidth networks, and RCP is a rate based protocol that improves the flow completion times for short flows. The idea of using router-centric explicit/precise feedback regarding available capacity to perform congestion control is not specific to the internet, and there exist proposals in the ATM network literature where mechanisms have been proposed for providing explicit and precise congestion feedback using the resource management (RM) cells for ABR (available bit rate) traffic ( [6], [61], [40] and [82]).

### 3.2 Congestion control in 802.11-based multi-hop wireless networks

The creation of the IEEE 802.11 standards [16] for a Carrier Sense Medium Access with Collision Avoidance (CSMA-CA) protocol, for single and multi-hop packet switched wireless networks, spawned research in the domain of multi-hop wireless packet switched networks. The success of
TCP, as a transport layer, in the wired Internet led researchers to experiment with the TCP/IP stack over 802.11-based wireless networks. This lead to the realization of the inadequacy of the congestion detection mechanism of TCP, leading to excessive false detection of congestion, resulting in the heavy underutilization of the network capacity [2, 18], or an unfair allocation of bandwidth to contending flows [68]. The shortcomings of TCP over wireless networks was primarily due to the assumption that a packet loss is a sufficient and necessary condition for congestion in the network. While packet loss is good indicator of congestion in wired networks [35], in wireless networks packet losses could occur due to buffer overflows (congestion), link errors, frequent route changes due to mobility, or collision due to interference [2, 18]. Thus, using packet loss as a sufficient signal of congestion can force TCP to woefully underperform in a wireless network.

Further, in wired networks a source can cause congestion only in router/nodes through which it’s routing its traffic; in wireless networks, however, the broadcast nature of the wireless medium allows sources to cause congestion even at nodes that are not routing their traffic. Thus, in a wireless network a flow might not incur packet loss and still might be responsible for causing congestion. Therefore treating packet losses as a necessary condition for congestion is incorrect.

The failure of the congestion detection mechanism in TCP, over wireless links, prompted researchers to explore new techniques for congestion detection/avoidance strategies that will improve upon the performance of TCP over wireless. One set of proposals (TCP-F [11], TCP-ELFN [32], TCP-BuS [44], ATCP [50], EPLN/BEAD [105]) that aimed at solving transport layer rate/congestion control in a multi-hop wireless setting, focused on improving the performance of TCP over wireless. They strove to achieve this objective by introducing mechanisms to better inform TCP of the exact reason for packet loss over a wireless link. By presenting the exact reason for packet loss to TCP, they could freeze the flow-control window in TCP when congestion
was not the cause of packet loss, and allow TCP’s default congestion avoidance mechanism to function when packet loss occurred due to buffer overflows.

While these proposals helped in solving the problem of treating packet loss as a sufficient signal of congestion, and improved network utilization, they still did not address the issue of unfairness introduced by treating packet loss as a necessary condition for congestion. To rectify the second problem, researchers have undertaken a neighborhood-centric view of congestion [67, 68], as compared to a host-centric view of congestion. By assuming packet loss (due to buffer overflows) as a necessary condition for a signal of congestion, TCP assumes a host-centric view, assuming that a source can cause congestion only in nodes/routers that are forwarding its data. In a neighborhood-centric view, congestion at a specific node/router is caused not only by the sources whose data the specific node is routing, but also by sources whose traffic is being routed by nodes within the neighborhood of the congested node. Link-RED [27] and NRED [103] are active queue management policies, similar to RED [25], that are implicitly (Link-RED) or explicitly (NRED) based on the design of the neighborhood-centric view of congestion.

Given the need for a neighborhood-centric view of congestion, and the failings of TCP in wireless, there has also been a significant push for clean slate design of congestion/rate control mechanisms over wireless networks, independent of TCP. Examples are ATP [84], EWCCP [87], WCP [68] and WCPCAP [68]. Of these, EWCCP, WCP and WCPCAP explicitly design congestion detection and rate allocation mechanisms that take the neighborhood-centric view into account. EWCCP is based on a cross-layer design principal, similar to the protocols proposed in this thesis, and aims to achieve proportional fairness amongst flows in a wireless ad-hoc network. EWCCP has however never been implemented in a real system. WCP [68] uses an AIMD mechanism to present a clean slate design of a transport protocol for wireless ad-hoc network. The
protocol is router-centric, and explicitly shares congestion amongst all neighbors of a congested node by asking all sources whose data is being routed through the congested nodes neighborhood to perform multiplicative decrease on their rates, on detecting congestion. By sharing the congestion signal equally amongst the nodes of a neighborhood it aims to achieve lexicographic max-min fair rate allocation amongst the flows. WCPCAP [68] is an explicit and precise rate control protocol designed for wireless ad-hoc networks. It uses the model proposed in [38] to calculate the exact available capacity in an 802.11-based wireless ad-hoc network, and shares this capacity equally amongst contending flows in order to achieve a lexicographic max-min fair rate.

### 3.3 Congestion control in wireless sensor networks

In the recent past Wireless Sensor Networks (WSN) emerged as a sub-domain of the research performed on multi-hop wireless networks. WSN share quite a bit of similarity with 802.11-based multi-hop wireless networks, in terms of the network structures and network dynamics; due the use of a wireless medium as the physical layer, and the use of Carrier Sense Medium Access as the de-facto MAC protocol in both type of networks. WSN are multi-hop wireless networks, composed of low rate, low powered, embedded devices having sensing capabilities. These networks are targeted towards sensing applications such as structural health monitoring [62], habitat monitoring [85], sensing seismic activity [99], etc. The need for congestion control has been demonstrated in wireless sensor networks [63, 67]. Given the importance of this problem, there have been a series of proposals aiming to mitigate the affects of congestion in a WSN. We summarize some of the key papers: ARC [101] proposes an AIMD rate control strategy whereby the
nodes increase rates proportional to the size of their sub tree and performs multiplicative decrease on sensing packet drops. ESRT [71] is a sink-centric approach that measures available capacity and allows for rate increments and decrements by observing the ability of sources achieve a certain event detection reliability target. CODA [93], provides for both open-loop hop-by-hop back-pressure and closed-loop multi-source regulation whereby sources vary their rate based on feedback from the sink. FUSION [34] uses a token based regulation scheme that allows for additive increase of source rates. It detects congestion using queue lengths and mitigates congestion by a combination of hop by hop back pressure and an adaptive MAC backoff scheme. In the work by Ee and Bajcsy [20], each node determines its own average aggregate outgoing rate as the inverse of its service time for packets and shares this rate equally amongst the nodes served in the subtree. This scheme does not always achieve a max-min fair solution as information is not shared explicitly with neighbors. IFRC [67] is a state of the art distributed approach that mitigates congestion by sharing congestion information explicitly with the set of all potential interferences of a node, and uses AIMD to react to the feedback. However, its design focuses primarily on achieving steady state fairness rather than rapid convergence or low delay. RCRT [63] is a recent centralized scheme where the sink employs an AIMD mechanism to calculate achievable rates and explicitly informs the sources as to the rate as which they can transmit. A common theme in most of these proposals is to use an AIMD mechanism to set rates, because they are agnostic to the underlying MAC and therefore lack fine-grained information regarding achievable network capacity.
3.4 Using cross layer optimization for rate control design

As can be seen from the progression of research in congestion control in wired and wireless networks, there has been a growing awareness that transport layer rate control functionality can be greatly improved by increasing the information exchange across layers (Transport, Routing and MAC). Further, there has also been a realization that transport layer rate control has primarily been an undertaking using heuristics. This has led to the pursuit of a more theoretically rigorous framework for quantifying the problem of transport layer rate control (e.g. FAST TCP [98]), and developing techniques that will result in algorithms which can achieve a cross-layer design.

While, in the wired Internet, rate control protocols developed using cross-layer design principles (such as XCP [41] and RCP [19]) have faced tempered skepticism — due to the incumbent nature of TCP in these networks, and the modifications required to intermediate routers — the adoption of these design principles for wireless networks has been more forthcoming. We proceed to highlight some of the work in this area, that have been responsible in developing and promoting the cross-layer design philosophy.

The 1998 paper by Kelly et al. [43] was a seminal one in the area of analysis of rate control in networks. It established that distributed additive increase-multiplicative decrease rate control protocols can be derived as solutions to an appropriately formulated optimization problem. The solutions can be interpreted as rate adaptation in response to shadow prices. This was followed by a classic work by Low and Lapsley [51] in which the Lagrange dual of the optimization problem is derived and a sub-gradient approach is used to solve the optimization problem in a distributed manner. Since then, there has been a substantial literature, primarily in the wired context, that has focused on understanding not only rate control but other network protocols across multiple
layers in the context of optimization problems that they seek to solve. This has led to the unifying view that layered and cross-layer protocol design can be understood fundamentally as the result of an appropriate decomposition of a network utility maximization problem. An elegant survey conveying this view of the literature is the recent article by Chiang et al. [14].

Given its compelling nature and power, it is hardly surprising that optimization approaches have been used for analysis and design of wireless network protocols as well. On the design side there have been a number of papers on cross-layer optimization in wireless networks: Of particular note are the authoritative papers by Chiang [13] and Johansson et al. [39], as well the works by Wang and Kar [95], Yuan et al. [106], Liu et al. [49], Mo et al. [53]. Many of these works focus on joint Transport and PHY/MAC optimization, and propose sub-gradient algorithms based on the primal as well as the Lagrange dual. In our own previous work in this direction [75], [76], we have focused on the particular problem of fair and efficient data-gathering on a tree and developed a Lagrange dual-based sub-gradient algorithm for it.

The past couple of years have also seen the emergence of new approaches that consider time-varying channels and provide for stochastic optimization. This approach is described by Neely [57], and expanded on in the monograph by Georgiadis [29]; another related work is by Xi and Yeh [102]. Approaches similar to the one proposed by Neely [57], have been proposed by Stolyar [83], Eryilmaz [21] and Lee [47]. These distributed approaches use neighborhood queue backlog occupancy information to make online transmission decisions. Connections between this stochastic optimization approach and the static optimization approaches are still being discovered, but it is already clear that these stochastic approaches are particularly useful for wireless networks showing a high level of link dynamics.
3.5 Backpressure-based rate control mechanisms

Recently, there has been an increasing effort to convert the theoretical frameworks proposed by Neely [57] and Stolyar [83], for solving the stochastic optimization problem of transport layer flow control using backpressure-based algorithms, into real-world network stacks. Most notable are the works by Warrier et al. [96], Akyol et al. [1] and Radunovic et al. [65]. The work by Warrier et al. [96] shows a practical mechanism of implementing backpressure-based, differential queue scheduling, in existing CSMA-based protocols. They however do not present any insights to the design of flow controllers that need to sit on top of their modified CSMA MAC for optimizing some global utility function. The work by Akyol et al. [1], apart from presenting a practical backpressure-based scheduling scheme on existing CSMA MAC's, similar to the work by Warrier et al. [96], also aim at building a flexible congestion control algorithm that can optimize any concave global utility function. The work by Akyol et al. [1] uses a primal/dual based greedy algorithm to solve the general optimization problem $P$ presented in chapter 1. The greedy algorithm emulates a gradient-based search and results in an additive increase style algorithm where the exponential time average rate is incremented based on the queue backlogs. The work by Radunovic et al. [65] develops a multipath routing and rate control protocol, that can be integrated with TCP over 802.11, using backpressure-based techniques. They use a simple backpressure scheduler that allows transmissions as long as the queue differential is greater than a threshold.
3.6 Capacity models for wireless networks

As shown in chapters 1 and 2, the main challenge in quantifying the transport layer rate control problem, as a constrained optimization problem, is to have a model that presents constraints which characterize the capacity region of the network [29]. A good survey of the models used in wireless networking research is presented in [53]. Among the most commonly used models are graph-based models, such as the clique capacity model ([12], [17]). The clique capacity model characterizes the capacity region of the network by presenting constraints associated with each clique existing in the network.

Researcher have also used link-based models, such as the slotted ALOHA model [94]. This model captures the link capacity between each sender-receiver pair as a success probability of a packet transmission on that link. The success probability on the link is calculated as the probability that the link is active, given that no other link between in the given neighborhood of the receiver is active. The constraints generated by the ALOHA model is simply that the rate allocated to each link should be less than the success probability on that link. The key drawback of this model is that it is very specific to the slotted ALOHA protocol. Apart from the graph-based and link-based models, researchers have also used a Signal to Noise Interference Ratio (SNIR) model [13]. In the SNIR model the link capacity is defined as the shannon capacity. In order to characterize the rate region of the network the model then sets a constraint on each link of the network, such that the link transmission rate should be less than the shannon capacity of the link.

Apart from the generic models described here, more recently there has been an effort to build accurate 802.11 CSMA MAC [16] specific models, to characterize the capacity region of an 802.11-based multi-hop network. These models are similar to the ALOHA model, in that they
aim at calculating the link capacity of each link in the network, similar to the ALOHA model. The
capacity of a link is equated to the success probability of a transmission between a sender-receiver
pair. The calculation of the success probability is done by performing a markovian analysis on
a state space transition diagram of an 802.11-based multi-hop network. The work by Jindal et
al. [38] is the most result in this effort.

3.7 Positioning of our contributions

The Wireless Rate Control Protocol (WRCP), presented in Chapter 5, is similar in spirit to
RCP [19] in its design and goals, since it is a rate-based protocol and attempts to shorten the
flow completion times by explicitly allocating rates based on available capacity in the network.
In an 802.11 multi-hop wireless setting, WCPCAP [68] is similar in nature to WRCP. WCPCAP
is a distributed rate control protocol that can achieve max-min fairness using explicit capacity
information. The key difference between WCPCAP and WRCP is that WCPCAP relies on a
model [38] that is very specific to an 802.11 multi-hop network. It is not clear how this model
can be ported to a sensor network setting. WRCP on the other uses a very simple model, that we
show works well in the context of a CSMA MAC for sensor networks. Further, WCPCAP does
not cater for external interference, or present validation for its parameter settings, whereas as we
will demonstrate, WRCP works well in the presence of external interference, and the parameter
settings for WRCP are better understood.

The Backpressure-based Rate Control Protocol presented in chapters 6 and 7 is our con-
tribution to convert the theoretical frameworks, resulting in backpressure-based algorithms for
solving stochastic optimization problem of transport layer rate control, into real world protocols.
In Chapter 6, we use the theoretical framework presented by Neely et al. [58] to design a flexible rate control protocol that can optimize for any application specific concave rate-utility function. Akyol et al. [1] presented a similar proposal to our work presented in Chapter 6 in the same time frame. The key difference between their work and our proposal (BRCP), is the stochastic optimization technique used in designing the congestion control algorithm, which are complimentary to each other making the congestion control algorithms complementary to each other as well.

The technique used by us relies on the Lyapunov drift framework presented by Neely [58]. This technique is complementary to the primal dual technique used in [1] and presents a purely queueing theoretic approach to solve the optimization problem $P$. Our proposal thus relies on setting instantaneous rates based on the queue backlog at a node, instead of using time averages and additive increase mechanisms used by Akyol et al. [1]. In Chapter 7 we add to this effort, of making backpressure-based protocols a reality, by undertaking a thorough empirical evaluation of the various design choices available to achieve an efficient implementation of a backpressure-based protocol. In Chapter 7, we show that in order to achieve the gains envisioned by implementing a backpressure-based protocol, at least in a sensor network setting, we do not need to implement the complicated heuristics presented by Warrier et al. [96] and Akyol et al. [1]. Instead we can have a naive scheduler that allows the network layer to inject packets into the MAC layer as long as the node has a positive queue differential with its parent, and achieve the same gains as claimed by Warrier et al. and Akyol et al.. In Chapter 7, we also show the parametric dependence of the backpressure-based stack, and motivate the argument for automatic parameter adaptation in order to ensure the performance gains promised by the backpressure-based designs. The empirical evaluation presented in Chapter 7 is the first of its kind, and raises some open problems not highlighted in previous proposals ( [65], [1], [96]).
Our design of WRCP and BRCP has been enabled by the use of the receiver capacity model (Chapter 4). It quantifies the capacity presented by a receiver to flows that are incident on the receiver, and presents constraints local to the receiver, that defines the bandwidth sharing that takes place between flows incident on a receiver. The model is particularly useful in our setting, since it caters to a CSMA based wireless sensor network. There are other interference models in the literature. Among the most commonly used models are graph based models, such as the clique capacity model ([12], [17]), and link based models such as the SINR model [13] and the ALOHA model [94]. We believe these models, which have been largely used in theoretical studies, are not well suited to CSMA-based networks. The clique capacity model does not present purely local constraints, making it hard to realize distributed algorithms using this model; the SINR model is more akin to modeling MAC’s with variable bit rate radios; the ALOHA model is very specific to the ALOHA MAC. The model presented by Jindal et al. [38] captures the capacity region of an 802.11 multi-hop wireless network quite accurately, however it is not clear how this model can be ported to be used in a wireless sensor network setting.
Chapter 4

Receiver Capacity Model\(^1\)

In Chapter 1, we showed that the problem of rate control can be written as a constrained optimization problem as follows:

\[
P : \max \sum_{r_i \in \mathcal{P}} g_i(r_i)
\]

\[s.t \quad \mathcal{R} \in \Lambda\]

For a given concave utility function \(g_i(r_i)\), the key challenge in quantifying the problem \(P1\) is to estimate the constraints of this optimization problem. The constraints of the problem \(P1\) are, in turn, defined by the capacity region \(\Lambda\) of the underlying network. As described in Chapter 1 (Section 1.2) the capacity region for the network is defined as the set of rate vectors that the network can support. Also, the boundary of the capacity region of a network depends on the underlying routing and MAC protocols employed in the network.

In order to quantify the problem \(P\) in the context of a wireless sensor network, we require a model that, for a given instance of a sensor network, can present a set of constraints which will characterize the capacity region of the network. The focus of this chapter is therefore to present such a model, that will present constraints which characterize the capacity region of a wireless

\(^1\)Proof of the feasibility of the receiver capacity model, presented in this chapter, was first published as [79].
sensor network; whose routing protocol results in a collection tree rooted at a sink, and which operates over a MAC that is a Carrier Sense Multiple Access (CSMA) based random access MAC. We will first show why it’s hard to design such a model that characterizes the capacity region of a CSMA-based multi-hop wireless network. We will then present an intuitive model, called the receiver capacity model, that gives us a good approximation of the capacity region for a CSMA-based multi-hop wireless sensor network over a collection tree.

4.1 Modeling the capacity region for a Wireless Sensor Network (WSN)

In order to understand the modeling of the capacity region for a wireless sensor network, it is beneficial to first understand the modeling of the capacity region of a wired network. To explain the model used in wired networks, we take a simple example of a 1-hop topology with two sources transmitting data to a single destination (Figure 4.1(a)), over a shared link. In a wired network, the link has a constant capacity. The capacity region for this simple wired network, when only two flows (source-destination pair) are active over the link, is shown in Figure 4.1. Given that the link capacity is constant, the capacity region turns out to be linear. The linear capacity region allows for a simple model that presents linear constraints over each link; indicating that all possible linear combination of flow rates can be supported over the link, as long as the sum of the flow rates does not exceed the link capacity.

It is evident that the above model, though described in the context of a simple 1-hop wired network, can be used to present constraints which characterize the capacity region for a multi-hop wired network. It can achieve this by presenting a similar linear constraint on each link in the
network. Together, these constraints will characterize the capacity region of the wired network. Characterizing the capacity region of a network as a set of local linear constraints helps greatly simplifying the design of bandwidth sharing/rate control algorithms; since, irrespective of the rate allocation amongst flows, the rate allocation is feasible as long as the sum rate is within a constant value.

In contention based multi-hop wireless networks, the boundary of the capacity region is not considered to be linear [92]. This can be seen in Figure 4.3, which gives the capacity region of a 1-hop wireless network, employing the slotted ALOHA protocol [4,86] for medium access, with two sources with backlogged queues (which implies that sources in the network always have a packet to send), transmitting data to a single sink over a shared medium. For this ALOHA-based wireless network [4] the lack of linearity results in the capacity being a function of the input source rates, making the design of rate control algorithms complex [95]. Making a linear approximation to the boundary of this capacity region can lead to a considerable loss, in terms of the achievable capacity, due to the concavity [7] exhibited by the boundary. The linear approximation will be strong for rates that are comparable to each other. However, the approximation can become weak as we move to the extremes, when only a single source is operational.

The reason for the difference in performance of the ALOHA medium access protocol, when sources have comparable rates, and when sources have disproportionate rate allocation is the lack of carrier sense in the ALOHA protocol. The lack of carrier sense results in high collision probability when sources have comparable rates, and low collision probability when sources have disproportionate rates. This observation can be interpreted in a different light to understand the behavior of the capacity region of an ALOHA network as the number of senders in the network are increased. The interpretation of the above observation can be that at least in a single hop
Figure 4.1: The capacity region of two flows over a single wired link.
Figure 4.2: A single-hop wireless topology.

network, with sources having backlogged queues, given a fixed number of sources, the center of the capacity region represents a scenario when all sources are active, while the extremes represent a scenario when only a few sources are active. For an ALOHA network, therefore, the linear approximation might hold for small number of sources, but will become worse as the number of the sources in the network increase.

Since the performance of the ALOHA network represents one extreme of the spectrum, the performance of a CSMA-based MAC (in terms of collision probabilities) should improve on the performance of ALOHA networks. Therefore, for the specific CSMA MAC that we will be dealing with, if we are able to show that the throughput achieved by sources does not degrade drastically as we increase the number of senders in the network; modeling the capacity region as a linear function might result in a good approximation of the true capacity region of the network. As seen in the case of a wired network, a model that presents us with local linear rate constraints to characterize the capacity region of the network might greatly simplifying the analysis of these
networks. It will present a simple quantification of the problem \( P1 \), allowing us to pursue a rigorous quantitative approach to transport layer rate control design for wireless sensor networks.

While presenting our model, it is important to remember that the insights learnt from the analysis of the simple ALOHA network were based on a capacity region obtained for a scenario where the sources are always backlogged. It has been shown that the throughput region for this scenario is a lower bound of the true capacity region that can be achieved in ALOHA networks [69, 86]. Therefore, we conjecture that the model presented in the succeeding sections, in theory, characterizes the lower bound of the achievable capacity region of a CSMA-based wireless sensor network. However, in practice, as will be seen in Chapter 5 the model works quite efficiently in presenting capacity to a rate control control algorithm, compared to a state of the art protocol that was not designed based on this model.

Figure 4.3: The capacity region of two flows in a broadcast domain of an ALOHA network.
4.2 The Receiver Capacity Model (RCM)

In order to simplify the modeling of the capacity region for a wireless sensor network, we propose the receiver capacity model. The crux of our approach is to associate the concept of capacity with nodes instead of links, we refer to this as receiver capacity. Each node can be perceived as having a receiver domain consisting of all transmitting nodes within range, including itself. This capacity is to be shared linearly by all flows traversing the corresponding receiver’s domain. Although in general the region of achievable rates in a given receiver domain is not linear, we approximate it with a linear rate region by making the receiver capacity a constant that depends only upon the number of neighboring nodes (not their rates). Although we believe on intuitive grounds that this is a good approximation for wireless sensor networks due to the small packet sizes (≈ 40 bytes) that are prevalent in such networks, we do not prove here rigorously how good this approximation is. We rely on the empirical results presented in our evaluation of WRCP (Chapter 5) which is designed based on this approximation, to show that it is extremely useful in practice.

In order to associate a value to the receiver capacity, we equate the capacity of a receiver with the saturation throughput of the CSMA MAC. The saturation throughput [5] of a CSMA MAC is defined as the throughput observed by the receiver when all senders are backlogged and are within each others interference range. While this does not cover all possible sender-configurations in a receiver domain, our experience with WRCP shows that although this estimate is potentially conservative, it leads to equal or better performance in terms of achievable goodput compared to the state of the art.

Our implementation is performed on the Tmote sky mote [55], which use the CC2420 radios, running TinyOS-2.0.2.2. Figure 4.4 presents the empirically measured saturation throughput for
Figure 4.4: Saturation throughput for multiple senders for the CC2420 CSMA MAC, with packet size of 40 bytes, on TinyOS-2.0.2.2

this platform as the number of senders in-range is varied. This graph allows us to associate a value with the receiver capacity for each node in a WSN collection tree.

Using the notion of receiver capacity, we can determine constraints on rate allocation to flows in a WSN collection tree. Let $N(i)$ be the set of all neighbors of $i$ (consisting of $i$ itself, all its immediate children, and all other nodes in its interference-range); $C(i)$ the set denoting the subtree rooted at $i$ (including itself); $r^i_{src}$ the rate at which data generated at source node $i$ is being transmitted; and $B^i$ the value of node $i$'s receiver capacity. The receiver capacity constraint at each node $i$ is then given as follows:

$$\sum_{j \in N(i)} \sum_{k \in C(j)} r^k_{src} \leq B^i$$ (4.1)

The set of all such constraints forms the receiver capacity model.
We explain this with an example. Figure 4.5 shows an 8 node topology. The solid lines indicate a parent-child relationship in the tree. The dashed lines are interference links. Rates indicated on interference links\(^2\) quantify the amount of interference generated by a neighboring node when it is transmitting data to its parent. Thus, when node 2 sends its data to node 1 at some rate, node 2 not only consumes the corresponding amount of capacity at node 1 but also at node 3; the rate label on interference link 2 → 3 is the same as that on link 2 → 1.

Based on our model, the constraint on the rates at node 3 would be as follows:

\[
r_{2\text{tot}} + r_{3\text{tot}} + r^6 \leq B^3
\]

(4.2)

where \(B^3\) is the receiver capacity of node 3 and \(r^6\) is the source rate of node 6. \(r_{2\text{tot}}, r_{3\text{tot}}\) and \(r_{7\text{tot}}\) are the output rates at node 2, node 3 and node 7 respectively and are given by:

\[
\begin{align*}
  r_{2\text{tot}} &= r^2 + r^4 + r^5 \\
  r_{3\text{tot}} &= r^3 + r^6 \\
  r_{7\text{tot}} &= r^7 + r^8
\end{align*}
\]

The half-duplex nature of the radios results in the term \(r^6\) appearing twice in equation 4.2. Once independently to account for the consumption of bandwidth during reception at node 3, and once as part of the term \(r_{3\text{tot}}\) to account for the forwarding of the flow originating at node 6.

\(^2\)For now, we assume links are perfectly bidirectional. In the WRCP protocol we will relax this assumption and handle lossy, asymmetric links
Figure 4.5: An illustrative example of the receiver capacity model

Based on equation 4.1, for Figure 4.5, we have the following constraints:

\[ r_{tot}^2 + r_{tot}^3 + r^4 + r^5 \leq B^2 \]  
\[ r^4 + r_{tot}^2 \leq B^4 \]  
\[ r^5 + r_{tot}^2 \leq B^5 \]  
\[ r^6 + r_{tot}^3 \leq B^6 \]  
\[ r_{tot}^7 + r^8 \leq B^7 \]  
\[ r^8 + r_{tot}^7 \leq B^8 \]  

4.3 Feasibility of RCM

The receiver capacity model (RCM), presented in section 4.2, is elegant and easy to use primarily due to the local constraints that the model presents. These local constraints allow for the protocols
developed using this model to be purely distributed. However, a key challenge in using the receiver capacity model, in practice, is to estimate the receiver capacity, $B_i$, of each receiver $i$ such that the rate vectors satisfying the local constraints are feasible. Feasibility of a rate vector implies that the rate vector satisfies the constraints of the model, and there exists a possible TDMA schedule that can satisfy the rate demand. In section 4.2, we mentioned that for a CSMA-based sensor network MAC it is sufficient to set the receiver capacity, $B_i$, to the saturation throughput of the CSMA MAC.

The equation of the saturation throughput of the CSMA MAC to the receiver capacity needs to be justified more rigorously since, from a theoretical standpoint, it is well known that if the local constraints in a model, such as the receiver capacity model, are set to the full physical layer link rate then it is possible to generate pathological topologies where the rate vector presented by the model becomes infeasible [23]. Infeasibility of a rate vector implies that the rate vector satisfies the constraints of the model, but there exists no possible TDMA schedule that can satisfy this rate demand. It is also well understood that in order to make the model present feasible rate vectors in every possible topology, it is imperative to set the constraints of the model to a fraction of the full physical link rate.

In this section, we present a sufficiency condition that will present us with the fraction of the physical link rate that the receiver capacity should be set to in order to guarantee its feasibility in every possible topology. Further, since the saturation throughput of the CSMA MAC is a fraction of the physical link rate, the sufficiency condition presented here can be used to justify our equation of the receiver capacity to the saturation throughput in a CSMA-based sensor network MAC. We present this justification in Section 4.4.
In this section, we first present a few examples, that show pathological scenarios where models that rely on local constraints will fail if the local constraints are set to the maximum possible link rate. To highlight the fact that this problem is not specific to the receiver capacity model, we present these examples using not only the receiver capacity model, but the Clique Capacity Model (CCM) [23] as well. The clique capacity model is also a graph based interference model, commonly used to quantify cross-layer optimization problems [17] in wireless networks. CCM characterizes the capacity region of the network by presenting constraints associated with each clique existing in the network. Given that CCM and RCM both present local constraints to characterize the capacity region of a wireless network, it is important for us to justify the use of RCM over CCM at this juncture. We propose RCM as the model of choice since, in practice, as will be seen in chapters 5 and 6 and the examples presented below, the receiver constraints generated by RCM can be realized in a real system by simply observing the neighbor table of a node in the network. In contrast, to use CCM, nodes will need to determine all the cliques in the network which is hard to achieve in a purely distributed manner. Thus, the simplicity of protocol design is much higher when using RCM as compared to CCM.

Following these examples we will proceed to present a proof that shows that if the constraints from the receiver capacity model are set to $\frac{1}{3}$ of the full link rate, it can be guaranteed that the receiver capacity model will present feasible rate vectors for all possible topologies. A similar bound exists for the clique capacity model as well. In [23] it is shown that the constraints of the clique capacity model guarantee a feasible rate vector for any wireless communication graph, only when they are set to $\frac{2}{3}$ of the full interference free link rate. It is important to understand that while the bound presented by CCM is higher than the bound presented by RCM, this does not necessarily imply that CCM presents a better capacity region than RCM. This is primarily
due to the fact that the constraints generated by CCM and RCM are different (see examples in Section 4.3.1), implying that the capacity regions presented by the two models are different. We consider an investigation into this question, wether the capacity region presented by one model supersedes the other, to be out of the scope of this thesis and hence it remains an open problem. Irrespective of an answer to this question, as mentioned earlier we consider RCM to be a better engineering choice than CCM due to its practical applications in developing distributed protocols.

4.3.1 Receiver Capacity Model vs Clique Capacity Model

RCM and CCM both present local constraints, and hence can be used to potentially develop distributed protocols/algorithms. As shown in the previous section, the constraints in RCM represent the capacity constraints on each receiver in the network. The constraints obtained from CCM, however are obtained using a conflict graph. The conflict graph is obtained by creating a new graph, replacing the edges in the original graph with vertices in the conflict graph, and connecting vertices in the conflict graph that interfere with each other. The constraints in CCM thus represent the constraints on edges that interfere with each other.

In this section we present two examples to highlight the following: first, since both models generate purely local constraints, if these constraints are set to the maximum link rate, there will exist pathological topologies where either of these models will fail. Both models thus present rate vectors that can capture only a subset of the true rate region. Second, though both models produce local constraints, it will be clear from these examples that it is much easier to realize these constraints in a real system using the receiver capacity model, rather than using the clique capacity model.
Figure 4.6: The pentagon topology. This topology helps explains the limitations of CCM.

4.3.1.1 Failings of CCM

The example topology presented in figure 4.6 highlights a particular scenario in which the clique capacity model fails. Figure 4.6(a) presents the network topology. The solid lines in this figure represent the transmitter receiver pairs, and the dashed lines represent the interference links. Figure 4.6(b) represent the conflict graph for the given network topology. The constraints generated for the above topology using the clique capacity model are as follows:

\[ r_A + r_B \leq 1 \]
\[ r_B + r_C \leq 1 \]
\[ r_C + r_D \leq 1 \]
\[ r_D + r_E \leq 1 \]
\[ r_E + r_A \leq 1 \]
For the above topology, the constraints generated using the *receiver capacity model* are as follows:

\[
\begin{align*}
    r_A + r_B + r_E & \leq 1 \\
    r_B + r_C + r_A & \leq 1 \\
    r_C + r_D + r_B & \leq 1 \\
    r_D + r_E + r_C & \leq 1 \\
    r_E + r_A + r_D & \leq 1
\end{align*}
\]

The max-min fair rate allocation achieved using the above constraints will be \( r_A = r_B = r_C = r_D = r_E = \frac{1}{2} \) for the clique capacity model, and \( r_A = r_B = r_C = r_D = r_E = \frac{1}{3} \) for the receiver capacity model. It can be easily seen that the rate allocation obtained by the clique capacity model is infeasible.

### 4.3.1.2 Failings of RCM

In the previous section, we showed a scenario where the clique capacity model failed. In this section we present a scenario (Figure 4.7) where the receiver capacity model fails. As mentioned earlier, these examples highlight the problem of setting local constraints the full link rate.

For the topology in figure 4.7(a) the constraints obtained using *CCM* are:

\[
\begin{align*}
    r_A + r_B + r_C & \leq 1 \\
    r_C + r_D & \leq 1
\end{align*}
\]
The constraints obtained using $RCM$ will be:

\[
\begin{align*}
  r_A + r_B & \leq 1 \\
  r_B + r_C & \leq 1 \\
  r_A + r_C & \leq 1 \\
  r_C + r_D & \leq 1
\end{align*}
\]

The max-min fair rate vector obtained using the constraints from $CCM$ is, $r_A = r_B = r_C = \frac{1}{3}, r_D = \frac{2}{3}$. However the max-min fair rate vector obtained using $RCM$ is, $r_A = r_B = r_C = r_D = \frac{1}{2}$. For this topology the max-min fair rate vector obtained by RCM is clearly infeasible.

In practice we have observed that if RCM constraints are set to the full link rate, the model fails usually when there are a large number of asymmetric links in the network.

The above are examples of pathological topologies, highlighting the deficiencies in both the clique capacity as well as the receiver capacity models. Apart from highlighting the failings of the two models, the above examples also highlight the the ease of using RCM over CCM. In the following section we present a sufficiency condition, which gives the fraction of the link rate to
which the receiver capacity need to be set in order to make RCM generate feasible rate vectors for all topologies.

### 4.3.2 Assumptions and definitions

In this section we present a quantitative description of a general communication graph and various terms pertaining to this graph that will be used in our proof for the sufficiency condition of the link rates for the receiver capacity model.

We represent the nodes in the network, and possible communication, with a directed labeled graph $G = (V, L, f, w)$ where $V$ represents the set of nodes in the network and $L$ the set of edges (links) in the network. Each link $l \in L$ is associated with two labels $f$ and $w$. The label $f(l)$ represents a demand rate at which data needs to be transmitted over the link $l$. The label $w(l)$ represents the number of slots that are required in order to transmit the data over the link at a rate $f(l)$.

Since the feasibility conditions are shown for a TDMA MAC, we assume the following about the underlying TDMA mechanism: the TDMA mechanism is assume to be composed of a frame of $T$ seconds, over which the demand $f(l) \forall l$ has to be satisfied. The schedule, designed to satisfy the demand $f(l) \forall l$, is repeated every frame. The frame of $T$ seconds consists of slots of duration $\tau$ seconds ($\tau < T$). A frame thus consists of $\frac{T}{\tau}$ slots. A feasible TDMA schedule will therefore allocate enough slots to each link $l$, satisfying its demand $f(l)$, such that no two interfering links have overlapping slot allocation.

For the graph $G$, it is assumed that only a subset of the link $l \in L$ are active. This subset is determined by the transmitter receiver pairs that want to exchange data. If $l$ is part of the set of links that are active then $f(l) \geq 0$, else $f(l) = 0$. It is assumed that $f(l) \forall l \in L$, are rational.
Assuming that the maximum link rate is 1 bit/sec, and the frame duration $T = 1$ sec; the value of $w(l)$, associated with each link $l \in L$, is determined as follows:

$$w(l) = \frac{f(l)}{\tau}$$

Where $\tau$ is some rational number. We can find a $w(l), \forall l \in L$, since $f(l)$ is assumed to be rational. A link requiring a rate of $f(l)$ will require $w(l) = \frac{f(l)}{\tau}$ slots per second to achieve this rate. There can be multiple such $\tau$ values that will give an integral value for all links $l \in L$, we choose the largest of these $\tau$ that makes $w(l)$ a positive integer $\forall l \in L$. Thus, every link in the graph $G$ will have two labels associated with it, a rate label $f(l)$ and a slot label $w(l)$.

Our objective is to determine a sufficiency condition such that, given a demand vector $f$ that satisfies the sufficiency condition, it is possible to design a TDMA schedule that can satisfy the demand $f$. Some other variables that are defined for the graph $G$ are as follows:

- $r(l)$ is defined as the receiver of a link $l \in L$.
- $t(l)$ is defined as the transmitter of a link $l \in L$.
- $N_i$, the set of neighbors of node $i$, such that for any $j \in N_i$ there exists a link $l \in L$ such that $r(l) = i, t(l) = j$ and $f(l) = 0$. Note that under the receiver capacity model this link will be an interference link.

We now define a few terms pertaining to the graph $G$.

**Definition 1.** An interference set $I(i)$ for a node is defined as $I(i) = \{l : r(l) = i, \text{ or } t(l) = i, \text{ or } t(l) = j, j \in N_i\}$. The set $I(i)$ thus contains the set of links that have $i$ as a receiver, the set of links that have $i$ as a transmitter, and the set of links that have a transmitter who is a
neighbor of $i$. The cardinality of the interference set $I(i)$ at a node $i$ represents the total number of terms in the receiver capacity constraint at node $i$, where $\alpha \leq 1$.

**Definition 2.** The graph $G$ $\alpha$-satisfies the receiver capacity model (RCM) if

$$\sum_{l \in I(i)} f(l) \leq \alpha, \forall i.$$ 

Note that this inequality represents the receiver capacity constraint at node $i$.

**Definition 3.** We define the contention graph $G_c$ for the graph $G$, by replacing every edge $l \in L$ with $w(l)$ vertices in $G_c$. The $w(l)$ vertices in $G_c$ form a clique. Beyond this, we add an edge between two vertices in $G_c$ if the edges of the corresponding vertices in $G$ interfere with each others transmissions. The vertices in the graph $G_c$ form the set $V_c$, and the edges in the graph $G_c$, form the set $E_c$. The maximum clique size in $G_c$ is denoted by $\Delta_c$.

**Definition 4.** We define the contention graph $G_{dc}$ as a directed version of the graph $G_c$. The set of vertices of the graph $G_{dc}$, $V_{dc} = V_c$. The set of edges $E_{dc}$ is a directed version of the edges in $E_c$. A directed edge $e_{dc} \in E_{dc}$, from $v_i \in V_{dc}$ to $v_j \in V_{dc}$ exists if the transmitter of the corresponding link $l_i \in L$ interferes with the reception of $l_j \in L$. A reverse directed edge will exist if the transmitter of $l_j \in L$ interferes with the reception of $l_i$.

**Definition 5.** We define the size of the maximum interference set, $\Delta_I$ for the graph $G$ as follows:

$$\Delta_I = \max_{\forall i \in V} \left( \sum_{l \in I(i)} w(l) \right)$$

### 4.3.3 The sufficiency condition

The answer to the question of the sufficiency condition which guarantees the schedulability of the demand vector $f$ is given by the following theorem:

**Theorem 1.** A graph $G$ that is $1/3$-satisfying RCM can be feasibly TDMA scheduled.
The proof of this theorem follows by combining the following theorems 2 and 3. We first present and prove them before proving this result.

**Theorem 2.** *For an \( \alpha \)-satisfying graph \( G \), for a given \( \tau \), we have at least \( \frac{\Delta_I}{\alpha} \) slots/frame.*

**Proof.** A graph \( G \) that is \( \alpha \)-satisfying is given by:

\[
\sum_{l \in I(i)} f(l) \leq \alpha, \forall i \Rightarrow \sum_{l \in I(i)} \frac{f(l)}{\tau} \leq \frac{\alpha}{\tau}, \forall i \Rightarrow \sum_{l \in I(i)} w(l) \leq \frac{\alpha}{\tau}, \forall i
\]

By definition 5 we have:

\[
\Delta_I \leq \frac{\alpha}{\tau} \Rightarrow \frac{\Delta_I}{\alpha} \leq \frac{1}{\tau}
\]

Since \( \frac{1}{\tau} \) represents the total number of slots available per frame, for the graph \( G \) we have at least \( \frac{\Delta_I}{\alpha} \) slots/frame.

**Theorem 3.** *The number of slots required to achieve a feasible schedule in the graph \( G \) is no greater than \( 3\Delta_I \).*

We will require lemma 1 and 2, presented below, to prove this theorem. We therefore present these lemmas before we prove this theorem.

**Lemma 1.** *The maximum clique size of the contention graph \( G_c \) is no more than twice the maximum interference set \( \Delta_I \) for the graph \( G \), i.e. \( \Delta_c \geq \frac{\Delta_I}{2} \).*

**Proof.** The interference set \( I(v_i) \), of a vertex \( v_i \in V \) in the graph \( G \) and the in-degree \( D_{v_j}^{in} \) for the vertex \( v_j \in V_{dc} \) of the directed contention graph \( G_{dc} \) such that \( r(e_j) = v_i, e_j \in V \), are related as follows:

\[
|I(v_i)| = D_{v_j}^{in} + 1
\]

57
Thus, $\Delta_I = \max_{v \in V_{dc}} (D_{v}^{in}) + 1$. We claim that $\max_{v \in V_{dc}} (D_{v}^{in}) + 1 \geq \frac{\Delta_c}{2}$. To prove this claim assume a contradiction $\max_{v \in V_{dc}} (D_{v}^{in}) + 1 < \frac{\Delta_c}{2}$. Consider the set of vertices forming the maximum clique in $G_c$ and call this set of vertices the set $C$. Since $V_{dc} = V_c$, the set $C$ exists in $G_{dc}$ as well. From our assumption, for the graph $G_{dc}$, the in-degree of each vertex $v \in C$ is less than $\frac{1}{2}n - 1$, where $\Delta_c = n$. Thus for these set of vertices the total number of incoming edges is less than $n(\frac{1}{2}n - 1)$ which is equal to $\frac{1}{2}n^2 - n$. However the total number of incoming edges, which is equal to the total number of edges for this set of vertices belonging to $C$ is $\frac{n^2}{2} - \frac{n}{2}$. Since $\frac{1}{2}n^2 - n < \frac{n^2}{2} - \frac{n}{2}$, it leads to contradiction. Hence $\max_{v \in V_{dc}} (D_{v}^{in}) + 1 \geq \frac{\Delta_c}{2}$ and hence $\Delta_I \geq \frac{\Delta_c}{2}$.

\[\square\]

**Lemma 2.** The number of slots required to achieve a feasible schedule in the graph $G$ is no greater than $\frac{3}{2}\Delta_c$.

*Proof.* We first define a graph $H$, such that edges on this graph are vertices on $G_c$ and nodes in this graph are edges on $G_c$. Since $G_c$ is the line graph of $H$, edge coloring of $H$ is vertex coloring of $G_c$. If $\Delta_H$ is the maximum node degree in $H$, then as shown by Shannon in [72] the maximum number of colors that will be required to edge color the graph $H$ will be $\frac{3}{2}\Delta_H$. However since $G_c$ is the line graph of $H$ by construction $\Delta_c \geq \Delta_H$ and hence $H$ can be edge colored in $\frac{3}{2}\Delta_c$ colors.

The bound on the number of colors required for vertex coloring of the graph $G_c$, gives us a bound on the number of slots required for a feasible schedule in $G$. This statement is true since no two vertices in $G_c$, which share an edge can have the same color in a feasible vertex coloring. Also, if two vertices in $G_c$ share an edge, the corresponding links in $G$ cannot be scheduled simultaneously. Therefore, interchanging a color for a slot, a feasible vertex coloring on $G_c$ also
represents a feasible TDMA schedule for $G$ satisfying a demand vector $f$. Thus, the number of slots required to achieve a feasible schedule in the graph $G$ is no greater than $\frac{3}{2} \Delta_c$. □

We can now prove theorem 3.

**Proof.** Theorem 3

By lemma 2, since the graph $G$ can be scheduled in $\frac{3}{2} \Delta_c$ slots we can guarantee a feasible schedule for the graph $G$ if $C \geq \frac{3}{2} \Delta_c$. Further, since by lemma 1, $\Delta_I \geq \frac{\Delta_c}{2}$, if $C \geq 3 \Delta_I$, we can still guarantee a feasible schedule for $G$. □

Given the proofs of theorems 2 and 3, we can now prove theorem 1.

**Proof.** Theorem 1

Theorem 3 states that if we have $3 \Delta_I$ slots, we can guarantee a schedule for the graph $G$. Theorem 2 states that for an $\alpha$-satisfying demand vector $f$ the minimum number of slots per frame is $\frac{\Delta_I}{\alpha}$. It can be easily seen that by setting $\alpha = \frac{1}{3}$, the minimum number of slots required to TDMA schedule a demand vector $f$ matches the minimum number of slots that are available for an $\alpha$-satisfying graph $G$. This implies that for $\alpha \leq \frac{1}{3}$ all demand rate vectors $f$, that are $\alpha$-satisfying, can be guaranteed a feasible TDMA schedule. This proves our main theorem. □

### 4.4 Justifying the use of the receiver capacity model

In Section 4.3 we showed that if the constraints of RCM are set to $\frac{1}{3} r d$ the maximum link rate, the model will guarantee feasible rates. An implication of this result is that the receiver capacity model presents only a fraction of the true rate region achievable with TDMA. However given
the practicality of the receiver capacity model, and in light of the results presented in [48] which show that the best distributed greedy strategy can achieve only 50% of the true rate region, these bounds become bearable. This result becomes even more relevant in the context of CSMA-based wireless sensor networks. In chapter 5 we will use the receiver capacity model to develop a rate control protocol that can achieve lexicographic max-min fairness over a collection tree. As mentioned in Section 4.2, while using this model with a CSMA-based sensor network MAC, we set the bound on the receiver capacity constraints equal to the saturation throughput of the CSMA MAC. Since the saturation throughput of the specific CSMA MAC under consideration (the CC2420 CSMA MAC [77]) is less than \( \frac{1}{3} \)rd the link rate, and the collisions in the wireless sensor network CSMA MAC are small (due the small packet sizes in these network ∼40 bytes), the CSMA MAC behaves as an inefficient TDMA MAC, allowing the receiver capacity model to accurately calculate the max-min fair rate. The analysis in section 4.3.3 thus presents a theoretical justification for the practical use of the receiver capacity model, which helps us in presenting an approximation to the rate region of a CSMA-based wireless sensor network.
Chapter 5

Designing an Explicit and Precise Rate Control Protocol

In Wireless Sensor Networks (WSN), the philosophy of performing congestion control has largely been based on router-centric approaches, which use explicit congestion feedback from intermediate nodes. The core mechanism for rate control used in existing proposals (ARC [101], CODA [93], FUSION [34], IFRC [67]) are based on additive increase multiplicative decrease (AIMD) algorithms. An AIMD-based scheme has the advantage that the protocol is agnostic to the underlying link layer, requiring no prior knowledge of the available capacity. This allows for modular protocol design.

Despite the benefits presented by the AIMD mechanism, a key drawback of AIMD-based rate control protocols is their long convergence time to the achievable rate, and long queue backlogs as the rates frequently exceed the available capacity (this is used as a signal from the network to indicate that it is time to cut back) [19]. This is illustrated in Figure 5.1, which presents the performance of IFRC [67], the state-of-the-art congestion control protocol in wireless sensor networks. These results are from a simple, single-hop, fully connected, 4-node experiment with 1 sink and 3 sources. It is observed that the rate allocation takes more than 300 seconds to

---

1This work was done in conjunction with Prof. Bhaskar Krishnamachair and was first published as [77].

61
converge, and queue sizes routinely reach 8-10 packets. The long convergence times do not affect the goodput of flows when the flows are long \textit{i.e.}, flows whose duration of activity is much longer than the convergence time and the number of flows in the network is constant (a \textit{static scenario}). However, we believe AIMD based rate control protocols will adversely affect the goodput of flows when the number of flows active in the system is continuously changing (a \textit{dynamic scenario}). Note that this will occur whenever there exist short flows in the network. In this scenario, the per-flow available capacity is continuously changing (due to the rapidly changing active flow count). If the long convergence time of the AIMD-based protocol prevents it from adapting to these changes fast enough, it is inevitable that active flows will be allocated sub-optimal rates. This sub-optimality has significant ramifications in terms of energy consumption, and hence on network lifetime. The lower the goodput, the longer it will take for the flows to complete, forcing radios in the network to be awake for a longer duration, and hence consuming more energy. Such scenarios are particularly relevant to event-driven sensor networks, and those that deploy duty cycling to conserve energy.

In this work, we aim to verify and address the above problems faced by AIMD-based rate control protocols. We focus on designing a distributed rate control protocol for WSN, one that will perform well not only in a static scenarios but in a dynamic scenario as well. We show that drastic improvements in the convergence time of a rate control protocol can be achieved if the protocol design is based on the knowledge of explicit capacity information, rather than on an AIMD mechanism. The key challenge in achieving this, of course, lies in overcoming the difficulty in computing the capacity, given that the bandwidth of each link is affected by interference from other links in its vicinity.
Our principal contribution in this work, for the specific case of a collection tree, is the design and implementation of a distributed rate control protocol, that we refer to as the *Wireless Rate Control Protocol* (WRCP). WRCP uses an approximation of the available capacity in order to provide explicit and precise feedback to sources. This approximation is obtained by exploiting performance knowledge of the underlying CSMA MAC protocol. The key idea in our approach is to associate a constant capacity with the nodes instead of the links. The gains of this approach, particularly in a dynamic flow setting, in terms of convergence times (few tens of seconds for WRCP as compared to hundreds of seconds for IFRC) and smaller queue backlogs are highlighted in Figure 5.2. The fast convergence times translate to higher goodput, and hence faster flow completion times (which indirectly results in energy savings), and the reduced queue size improves end-to-end packet delays.

![Figure 5.1: Rate allocation and queue backlog behavior for IFRC as observed at a particular node.](image)
The rest of the chapter is organized as follows. In Section 5.1, we present the software architecture used to design a rate control stack in the TinyOS-2.x operating system. In Section 5.2, we present the design and implementation of WRCP. This protocol has been designed to work specifically over a collection tree. It uses the receiver capacity (Chapter 4) model to provide explicit and precise rate control information to the sources, striving to achieve a lexicographic max-min fair allocation. In section 5.3, we present an analysis to estimate the parameter settings for WRCP that guarantees a stable operation for the protocol over any given topology. In Section 5.4 we present our experimental setup for evaluating WRCP on TinyOS-2.X, running on Tmote Sky devices, using the IEEE 802.15.4-compliant CC2420 radios. In Section 5.5 we present empirical evidence, justifying our analysis of parameter selection for WRCP. In Section 5.6 we undertake a comparative evaluation with IFRC [67]. The results show substantial improvements in flow completion times and end-to-end packet delays.
5.1 Software architecture for WRCP in TinyOS-2.x

Figure 5 shows the software architecture of WRCP implementation in TinyOS-2.0.2.2. TinyOS 2.0.2.2, already provides a framework for building collection trees in the form of the collection tree protocol (CTP) (TEP 123 [26]). Since WRCP aims to achieve lexicographic max-min fairness among sources over a collection tree, we have integrated WRCP with the collection tree protocol.

The only modification required to the existing MAC (the CC2420 CSMA MAC) was the addition of fields to the header to support rate control information and provide acknowledgements to broadcast packets using the software acknowledgement feature of TinyOS 2.0.2.2 (TEP 126 [54]). We also modified the routing engine in CTP to store additional per-neighbor information required by WRCP (current transmission rate, per-flow available capacity, current allocated per-flow rate, active flow state, number of descendants).

We also had to make a major modification to the forwarding engine of CTP, since the default forwarding engine of the collection tree protocol does not implement a FIFO queue. It implements a form of priority queuing which restricts the number of packets originating from an application in the node itself to one, giving higher priority to flows from its descendants. Since, our algorithm explicitly assumes that the forwarding engine treats all flows equally, we needed to implement a variant of the forwarding engine that implements a FIFO queue.

In our software architecture, the core functionality of WRCP is implemented in the Rate Controller. The Rate Controller performs the per-flow rate calculations, and sets the token generation rate on the leaky bucket [4]. The Flow Controller then uses tokens from the leaky bucket to admit packets into the system.
5.2 The Wireless Rate Control Protocol (WRCP)

While describing the receiver capacity model in Chapter 4, we assumed an idealized setting with constant-bit-rate static flows from backlogged sources and lossless links. A real world sensor network on the other hand would have asynchronous communication, lossy links and dynamic flows which might result from the on-demand nature of the sensing application. To implement the algorithm in a practical setting, we need to relax these assumptions. To this end we have designed the **Wireless Rate Control Protocol** (WRCP) which incorporates a number of mechanisms to handle these real-world concerns.

The objective of WRCP is to achieve *lexicographic max-min* fairness [4] over a collection tree. A lexicographic max-min fair rate vector $\mathbf{r}^*$ is a rate vector, such that if the elements of the max-min fair rate vector $\mathbf{r}^*$ are arranged in ascending order, for any other feasible rate vector $\mathbf{r}$ whose elements are also arranged in ascending order, $\mathbf{r}^*$ will always be lexicographically greater than $\mathbf{r}$, *i.e.* there exists a $j$, such that $r^*_j > \bar{r}_j$, and $\forall i$, $0 < i < j$, $r^*_i = \bar{r}_i$. In WRCP, we
achieve lexicographic max-min fairness by using a simple idea, that every receiver, at every time step, divides the available capacity at the receiver equally amongst all flows that are consuming capacity at the receiver.

5.2.1 The WRCP algorithm

In Figure 5.4, a single time step execution of the protocol is presented in the form of the WRCP algorithm. The variables used in the description of the WRCP algorithm are given in Table 5.1.

We elaborate on the definition of notation used in the description of the WRCP algorithm here; $P_i$ is the parent of node $i$; $r_i$ is the maximum allocated rate at which flows consuming capacity at node $i$ can operate; $r_i^{\text{tot}}$ is total transmission rate of node $i$. It is important to note the the transmission rate $r_i^{\text{tot}}$ is different from the source rate $r_i$. The rate $r_i^{\text{tot}}$ is the rate at which a node is forwarding packets (both packets generated at node $i$, and packets routed through node $i$), while $r_i$ is the rate at which a source at node $i$ can inject packets into the system. $r_i^{\text{ext}}$ is the amount of external interference experienced by a node $i$. $\gamma_i$ is the per-flow available capacity at node $i$. This is the maximum capacity by which sources consuming capacity at node $i$ can increment their rates. $p_{ji}$ is the link success probability between sender $j$ and receiver $i$. $N_i$ is the set of neighbors of node $i$ (including its one hop children). The set $N_i$ can change over time, depending on whether a neighbor is active and has a flow to forward. $F_i$ is the total number of flows being forwarded by a node $i$.  

67
Table 5.1: Variables used in the WRCP algorithm formulation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_i$</td>
<td>The set of neighbors of node $i$</td>
</tr>
<tr>
<td>$P_i$</td>
<td>The parent of node $i$</td>
</tr>
<tr>
<td>$T$</td>
<td>The rate update interval</td>
</tr>
<tr>
<td>$B_i$</td>
<td>The receiver capacity</td>
</tr>
<tr>
<td>$\gamma_i$</td>
<td>The per-flow available capacity at node $i$</td>
</tr>
<tr>
<td>$F_i$</td>
<td>The total number of flows being forwarded by $i$</td>
</tr>
<tr>
<td>$F_i^t$</td>
<td>The number of active flows in $i$’s neighborhood</td>
</tr>
<tr>
<td>$r_{i}^{tot}$</td>
<td>The total transmission rate of node $i$</td>
</tr>
<tr>
<td>$p_{ji}$</td>
<td>Link quality between $j$ and $i$</td>
</tr>
<tr>
<td>$r_{i}^{ext}$</td>
<td>The amount of external interference seen by $i$</td>
</tr>
</tbody>
</table>

WRCP makes rate update decisions at an interval of $T$ seconds. In step 1, WRCP first calculates the available per-flow capacity at node $i$, using the following equation:

$$\gamma_i((n+1)T) = \frac{B_i(nT) - r_i^{ext}(nT) - \sum_{j \in N_i(nT)} p_{ji} r_j^{tot}(nT)}{\sum_{j \in N_i(nT)} p_{ji} F_j(nT)} \quad (5.1)$$

In step 3 it then calculates the minimum per-flow available capacity at this node, by comparing its per-flow available capacity (step 1) with the per-flow available capacity it has overheard from all its neighbors. All flows that are consuming capacity at this node can increment their rates by at most this value ($\gamma_i^{min}$). Equation (5.1) captures the essence of the simple idea described above, allowing WRCP to achieve max-min fairness. The numerator in equation (5.1), is simply the remaining capacity at node $i$, and the denominator is the total number of flows that are consuming capacity at $i$. Equation (5.1) therefore simply distributes the available capacity, amongst all contending flows equally.

In steps 5-21, WRCP updates the current source rate of any source that is operating at that node. The way the rate update is performed depends on whether the minimum available capacity calculated in step 3 is positive or not. If it is negative, and the negative per-flow available capacity
has been learnt from another node, than the advertising node is considered to be a “bottleneck node” (steps 9-14), and the current source rate is set to source rate advertised by the bottleneck node if the source rate of the node is greater than that advertised. If however, the minimum per-flow available capacity is positive, or if the node itself is the “bottleneck node”, the source rate is updated using the following equation (step 18):

$$r_i((n+1)T) = r_i(nT) + \alpha \times \gamma_i^{\min}((n+1)T)$$  \hspace{1cm} (5.2)

where $\alpha$ is a constant, and as will be seen in Section 5.3 is responsible for the stability of the protocol. The need for $\alpha$ arises due to the lack of accurate time synchronization, which along with multi-hop communication can lead to inherent delays in the dissemination of rate control information, specifically the available capacity, across the network. This can lead to nodes constantly exceeding capacity, getting delayed feedback on congestion and reducing their rates by the excess capacity, exhibiting an oscillatory behavior. To dampen the oscillations, we introduce a coefficient, $\alpha \leq 1$, into the rate update equation. A small value of $\alpha$ ensures nodes acquire available capacity slowly, allowing for convergence.

In the remainder of this section, we describe different mechanisms implemented as part of the WRCP protocol in order to calculate various information such as the update interval $T$, the total number of active flows $F_i$ consuming capacity at node $i$, the sender link quality $p_{ji}$, and the amount of external interference ($r_i^{ext}$) that are used as inputs to the WRCP algorithm. These mechanisms guarantee that WRCP is able to calculate the required information in a realistic wireless sensor network setting.
Algorithm **WRCP**

1. Calculate Per Flow Available Capacity:
   
   \[ \gamma_i((n+1)T) = \frac{B_i(nT) - r_i(nT)}{\sum_{j \in N_i(nT) \cup P_i} r_j(nT)} \]

2. Calculate minimum available per flow capacity:
   
   \[ \gamma_i^{\text{min}}((n+1)T) = \min(\min_{j \in N_i(nT)} \gamma_j(nT), \gamma_{P_i}^{\text{min}}) \]

3. Update per flow rate:
   
   if \( \gamma_i^{\text{min}}((n+1)T) < 0 \) and \( \arg(\gamma_i^{\text{min}}((n+1)T)) \neq i \)

   then \( k = \arg(\gamma_i^{\text{min}}((n+1)T)) \)

   if \( r_i > r_k \)

   then \( r_i = r_k \)

   if \( r_i > r_{P_i} \)

   then \( r_i = r_{P_i} \)

   return

4. if \( r_i > r_{P_i} \)

   then \( r_i = r_{P_i} \)

5. Broadcast to \( j \in N_i: \)

   \[ \gamma_i((n+1)T), \gamma_i^{\text{min}}((n+1)T), r_i((n+1)T) \]

---

**Figure 5.4**: A single time step execution of the WRCP protocol at node \( i \).
5.2.2 Rate update interval

WRCP relies on a $T$ second timer to present nodes with an interval to calculate their rate updates. In order to maintain a fair rate allocation, and system stability, it is essential that $T$ be large enough to guarantee that rate control information has propagated to all nodes in the network within this update interval $T$. The value of $T$ depends on the depth, and quality of links for a given collection tree.

Traditionally transport protocols on the Internet, such as TCP, XCP [41] and RCP [19], rely on the end-to-end reliability built into the protocols to obtain an RTT estimate for each source in the network. They then use this RTT estimate to determine the rate update interval. WRCP, similar to existing rate control protocols ([63, 67]) for sensor networks, however does not have an end-to-end reliability mechanism. Hence WRCP needs to explicitly implement a mechanism to estimate this update interval for a given topology.

The mechanism implemented is as follows: whenever the root generates a control packet (In a collection tree the root consumes all data packets and hence has to explicitly generate a control packet) it associates a control sequence number with this packet. The control sequence number is added to MAC header before broadcasting the packet. The root increments the control sequence number by one, if and only if it has received an acknowledgement from all its 1-hop children. A node sends acknowledgement to a specific control sequence number as follows: if a node is a leaf node, it acknowledges every control packet it gets, by setting a control sequence acknowledgement field in the MAC head of all its outgoing data packets. A parent, if it sees the control sequence acknowledgement field set on receiving a packet from its child, will set the control sequence
acknowledgement field in the MAC header of its data packets if it has received an acknowledgement from all its 1-hop children. In this manner, control sequence number acknowledgement gets aggregated at root of each sub-tree, and flows up the collection tree. The root will receive a control sequence acknowledgement for its current control sequence number, when all its 1-hop children received an acknowledgement from their respective sub-trees. In order to estimate the rate update interval $T$, the root will keep a running estimate of the time taken to increment the control sequence numbers. It will then propagate this estimate throughout the network, to keep rate updates in sync for all nodes.

5.2.3 Estimating receiver capacity

As mentioned earlier, we approximate the receiver capacity by the saturation throughput, which is a function of the number of senders in the receiver’s range. The saturation throughput is pre-calculated and stored as a lookup table. Figure 4.4 shows that the saturation throughput will almost remain a constant as long as the number of senders is greater than 4.

5.2.4 Estimating neighborhood active flow counts

To calculate the available per-flow capacity (equation (5.1)), WRCP requires the number of active flows at a receiver. In a dynamic environment, the number of active neighbors and hence the number of active flows, in a given neighborhood is not constant. To handle the ephemeral flow count, an active flow state tag is associated with each neighbor entry. Aging this entry in the absence of packets from the neighbor helps give a good estimate of active flows in the network. The number of flows in a neighborhood determine the per flow available capacity at a receiver. A conservative estimate can be obtained by simply looking up the total number of active sources
that each neighbor is forwarding, without regard for the link quality with the neighbor. However, recent empirical results have shown that capture effects are quite dominant in these networks [74]. These results suggest that nodes with stronger links will cause more interference (or consume more capacity) than nodes with weaker links. We therefore take an optimistic approach and weigh the number of flows from a sender $j$ to a receiver $i$ by its link quality $p_{ji} \in [0, 1]$. The active flow count at a node $i$ is thus given by the following equation:

$$F_i = \sum_{j \in N_i} p_{ji} F_j(nT)$$  \hspace{1cm} (5.3)

Where $F_j$ is the number of flows being forwarded by a node $j$.

### 5.2.5 Estimating transmission rates

Another term required in the calculation of the available per-flow capacity (equation (5.1)) at a node $i$, is the current transmission rate $r_{i}^{tot}$ of each neighbor. To cater for non-CBR traffic, we maintain an exponential weighted moving average of transmission rates as follows:

$$r_{i}^{tot} = (1 - \beta) r_{i}^{tot} + \beta \frac{P_{kts \, Transmitted}}{1 \, sec}$$  \hspace{1cm} (5.4)

$P_{kts \, Transmitted}$ are the total number of packets\(^2\) sent out in 1 second, including retransmissions. Thus, the exponential moving average of the transmission rate is computed every second. Incorporating retransmissions into the calculations implicitly incorporates the affects of link losses into per flow available capacity calculations, since retransmissions due to link losses will result in a higher transmission rate forcing the receiver to advertise a smaller available capacity.

\[^2\]These packets include those originated at this node, as well as those being forwarded by this node.
An important point to be noted in equation (5.1) is that, as with the estimation of the active flow counts, the transmission rate used by a node $j$ is also weighed by the link quality from $j$ to $i$. The argument for weighing the transmission rate by the link quality while estimating the remaining available capacity is the same as that used for calculating the active flow count (Section 5.2.4).

5.2.6 Estimating sender link quality

WRCP needs the link quality between a sender and the receiver in order to estimate the per-flow available capacity (equation(5.1)). WRCP requires link quality estimate only in a single direction (from the sender to the receiver), simplifying the link estimation. Since every node is operating in promiscuous mode, the forwarding engine of node $i$ maintains a variable $rcv_{ji}$, which count the total number of packets received from a sender $j$, for the last 10 packets that the sender had transmitted. Once it observes that the sender has sent out 10 packets, (which the receiver realizes with the help of a transmission counter that the sender sends along as part of the MAC header of data packets) the receiver updates the moving average estimate from a particular sender $j$ as follows:

$$p_{ji} = \beta p_{ji} + (1 - \beta) \frac{rcv_{ji}}{10}$$

(5.5)

After updating the link quality $p_{ji}$, the receiver resets the receiver counter to $rcv_{ji} = 1$.

5.2.7 Estimating external interference

IEEE 802.15.4, the de-facto standard used for MAC protocols in sensor networks, suffers from severe external interference by 802.11 networks due to spectrum overlap. Given the ubiquitous presence of 802.11 networks, it is imperative that any rate control protocol for WSN have the
ability to predict the amount of external interference and incorporate this estimate in deciding
the sustainable rates. WRCP predicts the amount of external interference by observing the queue
size at a node. We believe the queue size represents a good estimate of the external interference,
since the only case when WRCP rate predictions can go wrong is in the presence of external
interference (since the receiver capacity model does not take external interference into account).

To estimate the amount of external interference to be used in WRCP’s rate calculations, we there-
fore use the following mechanism; every node $i$ maintains an exponential moving average of its
forwarding queue size $U_i$. The external interference experienced by a node $i$ is then given by the
following equation:

$$ r_i^{ext} = (1 - \beta) r_i^{ext} + \beta \frac{U_i}{1 \text{ sec}} $$

(5.6)

As is evident, the above moving average is updated every 1 second. The external interference,
along with the transmission rates of the neighbors (as well as the nodes own transmission rate)
are used to calculate the available per-flow capacity, described in the next section.

5.2.8 Rate bootstrapping for flow joins

If WRCP were to use equation 5.2 when a flow $i$ joins the network, flow $i$ might never get to
increment its rate (or it might receive unfair rate allocation) if the network has been operational
for a while, resulting in complete consumption of the network capacity. In such a scenario the
new flow will see $\gamma_i^{min}((n + 1)T) = 0$, not allowing the rate $r_i$ to increment. In order to allow
WRCP to adapt to flow dynamics, we use a bootstrapping mechanism in which a new flow $i$
enters the network in a phase called the *bootstrap* phase. In the *bootstrap* phase, a flow $i$ joining the network uses equation 5.2 if $\gamma_{i}^{\text{min}} > 0$, else it uses the following rate update equation:

$$r_{i}((n+1)T) = 2 \times r_{i}(nT)$$  \hspace{1cm} (5.7)

The *bootstrap* phase allows the new flow to increment its rate even if the remaining capacity has become negative. If the remaining network capacity is negative, this will force existing flows $j$ to reduce their rates. The *bootstrap* phase for a flow $i$ ends when its rate exceeds the per flow rate of the bottleneck node, while the remaining available capacity is still negative, i.e. when $\gamma_{i}^{\text{min}}((n+1)T) < 0$, and $r_{i}((nT)) > r_{k}$, where $k$ is the *bottleneck* node. The end of the *bootstrap* phase indicates that the new flow $i$, has forced the existing flows $j$ to reduce their rates making flow $i$'s rate equal to its bottleneck flow rate.

### 5.3 Parameter selection

The performance of WRCP, in terms of its stability, depends primarily on the parameters $\alpha$ and $T$. As has been described in section 5.2.2, the parameter $T$ is determined dynamically by WRCP based on the topology and the routing tree. The mechanism used to determine the parameter $T$ also ensures that the rate update interval for all sources in the network is homogeneous. Thus, the only tunable parameter required to guarantee the stability of WRCP is $\alpha$. In this section, we present an analysis to determine bounds on the parameter $\alpha$ that will guarantee a stable operation for WRCP.
The rate update equation used by WRCP is given by equation 5.2. If $B_i$ is the receiver capacity at a bottleneck node $i$, the term $\gamma_{i}^{\text{min}}$ can be given by:

$$\gamma_{i}^{\text{min}}((n + 1)T) = \left(\frac{B_i(nT) - \sum_{j \in C_i} r_j(nT)\Gamma_j - \sum_{g \in N_i, k \in C_g} r_k(nT)\Gamma_g}{F_i}\right)$$

Here $r_i(nT)$ is the per flow rate at source $i$, $C_i$ is the set of children under the subtree of node $i$, $N_i$ are the neighbors of node $i$, $\Gamma_j$ is the expected number of transmissions required to successfully send a packet from a node $j$ to its parent, and $F_i$ is the total number of flows that are consuming bandwidth at node $i$. Effectively, the second term in equation (5.2) has been replaced in the above equation, by a term that represents the remaining available capacity at bottleneck node $i$.

The CSMA MAC uses a stop and wait policy to ensure reliability at the data link layer. The term $\Gamma_i$, the number of packet transmissions required to ensure that a packet successfully transmitted from a node $i$ to its parent, can be calculated using the following recursive equation:

$$\Gamma_i = (1 + \Gamma_i)(1 - p_i^f) + (2 + \Gamma_i)p_i^f(1 - p_i^r) + 2p_i^f p_i^r$$

Where $p_i^f$ is the probability of successfully transmitting a packet from a node $i$ to its parent, and $p_i^r$ is the probability of successfully receiving an ACK from the parent. Solving the recursive equation, we can represent $\Gamma_i$ in terms of the link quality $p_i^f$ and $p_i^r$ as follows:

$$\Gamma_i = \frac{1 + p_i^f}{p_i^f p_i^r}$$
We now present some assumptions which help simplify our analysis. We replace the term $\Gamma_j$ for each $j$, by $\Gamma_{avg}$ where $\Gamma_{avg}$ is the average packet transmissions between parent and a child for the entire network. For this analysis we assume nodes have perfect synchronization, and accurate information allowing all flows at the bottleneck node $i$ to have the same per flow rate at any given instant of time. Thus, $r_k(nT) = r_i(nT)$, for each flow $k$ that consumes capacity at node $i$. Also the number of active flows is assumed to be a constant in the network making $B_i(nT) = B_i$.

Equation 5.2 can be rewritten as:

$$r_i((n+1)T) = r_i(nT) + \alpha \times \left(\frac{B_i}{F_i} - r_i(nT)\Gamma_{avg}\right)$$

(5.8)

Based on equation 5.8 we state the following Lemma:

**Lemma 3.** A rate control protocol using equation (5.8) will converge if:

$$0 < \alpha < \frac{2}{\Gamma_{avg}}$$

**Proof.** Assuming that all sources start with minimum constant rate $r_i(0)$, due to the recursive nature of equation 5.8, we can rewrite equation 5.8 in terms of $r_i(0)$, $B_i$, $p_{avg}$, $\alpha$ and $n$ as follows:

$$r_i(nT) = r_i(0)(1 - \alpha\Gamma_{avg})^n + \alpha\frac{B_i}{F_i}\left(\sum_{k=0}^{n-1} (1 - \alpha\Gamma_{avg})^k\right)$$

(5.9)

Thus,

$$r_i(nT) = r_i(0)(1 - \alpha\Gamma_{avg})^n + \frac{B_i}{\Gamma_{avg} F_i}\left(1 - (1 - \alpha\Gamma_{avg})^n\right)$$
For $r_i(nT)$ to converge as $n \to \infty$, $0 < \alpha \Gamma_{avg} < 2$. Thus, for WRCP to converge it is essential that $0 < \alpha < \frac{2}{\Gamma_{avg}}$.

5.4 Experimental setup

Our implementation is performed on the Tmote sky motes, which have CC2420 radios, running TinyOS-2.0.2.2. The experiments are conducted over 20-node (Figure 5.5) and 40-node (Figure 5.6) topologies on the USC TutorNet testbed [46]. Experiments were conducted over a period of few months to ascertain any change in the performance of the protocols due to link quality variations in the topologies. The average link quality ranged from 30%-75%. It was observed that the link quality variations for the topologies over this large time frame were negligible, lending validity to the results. The experiments were conducted on channel 26 of 802.15.4 standards.
The experiments were conducted on this specific channel to have an external interference free environment (this is the only channel in 802.15.4 that does not overlap with 802.11 channels), allowing us to present reproducible results. In order to show that the protocol shows good performance on channels suffering from external interference as well, in Section 5.6.6 we present WRCP performance results in the presence of external interference. In all experiments, sources are generating *Constant Bit Rate* (CBR) traffic.

![Figure 5.6: A 40-node topology.](image)

### 5.5 Stability analysis

In this section we present empirical evidence to validate the analysis presented in section 5.3, which is used in estimating the parameter settings for WRCP. In section 5.3 we had shown that for a given topology as long as \( \alpha < \frac{2}{\Gamma_{avg}} \), where \( \Gamma_{avg} \) is the average number of transmissions between
a node and its parent, WRCP will remain stable. In order to empirically justify this statement we ran WRCP on the two topologies shown in figures 5.5 and 5.6. For each of the topologies, we varied the value of $\alpha$ from 0.05 to 1.0 and observed different metrics of performance for WRCP. The observed metrics were the variance of the allocated rate and the average end-to-end packet delay observed amongst all packets received at the base station. For values of $\alpha$ for which WRCP is stable, the variance of the allocated rate should be small. For each of the topologies, these experiments were performed at two different power levels, at a power level of 5 and a power level of 10. For each of the two topologies Table 5.2 presents the estimated values of $\Gamma_{avg}$ and the bound on $\alpha$, measured at different power levels.

Table 5.2: The change in theoretical bound of $\alpha$, with change in topology.

<table>
<thead>
<tr>
<th>Topology</th>
<th>$\Gamma_{avg}$</th>
<th>$\frac{2}{\Gamma_{avg}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-node, Power = 5</td>
<td>4.2</td>
<td>0.476</td>
</tr>
<tr>
<td>20-node, Power = 10</td>
<td>5.61</td>
<td>0.3565</td>
</tr>
<tr>
<td>40-node, Power = 5</td>
<td>7.35</td>
<td>0.2721</td>
</tr>
<tr>
<td>40-node, Power = 10</td>
<td>12.05</td>
<td>0.1659</td>
</tr>
</tbody>
</table>

Figure 5.7: Evaluating behavior of WRCP with $\alpha$ for the 20-node topology.
As can be seen from the figures 5.7(a), and 5.8(a) the variance in the allocated rate rises quite sharply once \( \alpha \) becomes greater than the corresponding value of \( \frac{2}{\Gamma_{avg}} \), presented in table 5.2. Increase in the variance indicates that as \( \alpha \rightarrow \frac{2}{\Gamma_{avg}} \) the system takes a longer time to converge, and once \( \alpha > \frac{2}{\Gamma_{avg}} \) the variance becomes large, implying oscillations. This behavior is observed for the delay metric as well. The delay increases as \( \alpha \rightarrow \frac{2}{\Gamma_{avg}} \), and for values of \( \alpha > \frac{2}{\Gamma_{avg}} \) the delay is higher than the delay when \( \alpha < \frac{2}{\Gamma_{avg}} \). The increase in delay, for large values of \( \alpha \), is primarily due to the delayed feedback in the system. The delayed feedback results in nodes having stale information for their rate updates, causing erroneous increments and decrements. These erroneous increments regularly force the system to operate beyond the sustainable region, resulting in large queues and hence longer delays.

![Graphs](image.png)

**Figure 5.8:** Evaluating behavior of WRCP with \( \alpha \) for the 40-node topology.

The empirical evidence presented here validates our estimates for \( \alpha \), and proves that as long as \( \alpha < \frac{2}{\Gamma_{avg}} \) the system remains stable.
5.6 Comparative evaluation

In this section we present a comparative evaluation of WRCP with the Interference Aware Rate Control protocol (IFRC) [67], the state-of-the-art AIMD mechanism for rate control over a collection tree. The comparison of WRCP with IFRC highlights the advantages of having an explicit capacity based rate control protocol, as compared to one based on an AIMD mechanism, especially in a dynamic flow scenarios.

5.6.1 IFRC

Rate allocation in IFRC is split into two phases. When a source joins the network it starts in the slow-start phase. The slow start phase is similar to the slow-start in TCP. In this phase a source starts with a small initial allocated rate ($r_{init}$) and increases its allocated rate by a constant amount $\phi$ every $1/r_i$ seconds, where $r_i$ is the current allocated rate of the source. Thus, in the slow start phase, at every step the source increments its rate by $\phi \times r_i$, leading to an exponential increase. The slow-start phase continues till the source detects its first congestion event (average queues exceed a certain threshold). At this point, the source enters the additive increase phase. In the additive increase phase the source starts with an initial value of $r_{Thresh} = \frac{r_i(t_{cong})}{2}$, where $r_i(t_{cong})$ is the source rate at the last congestion event. In the additive increase phase, a source increments its rate by $\delta r_i$ every $1/r_i$ seconds, leading to a constant increment of $\delta$ at every step. The details of each of these mechanisms and the methodology for parameter selection is given in [67].

As will be seen in our evaluation, for IFRC the speed of convergence, in terms of allocating the achievable max-min rate, to each source in the slow-start as well as the additive increase
phase, depends on the initial values ($r_{init}$ for slow-start and $r_i(t_{cong})$ for additive increase) and the maximum achievable rate.

IFRC was originally implemented over TinyOS-1.x. On performing experiments with the 1.x stack we observed a considerable difference between the throughput achieved by IFRC on 1.x [67] and WRCP in 2.0.2.2. The gap was due to the performance difference between the communication stack of 1.x, that had to be modified for enabling software ACK’s required by IFRC, and the communication stack of 2.0.2.2. In order to have a fair comparison between WRCP and IFRC, we decided to port IFRC to TinyOS-2.x. In order to validate the porting of IFRC from TinyOS-1.x to 2.0.2.2, the behavior of the allocated rates observed in TinyOS-2.0.2.2 was compared to the ones presented in [67] and found to be the same, and performance of IFRC over TinyOS-2.x was found to be better than the performance of IFRC over TinyOS-1.x.

![Figure 5.9: Rate allocation for 20-node static case.](image)
For all experimental results presented in this section, the size of the payload was 10 bytes. WRCP adds 16 bytes, where as IFRC adds 26 bytes of overhead to each packet. Since both protocols exchange control information over the data path using a promiscuous mode of operation, WRCP exhibits better overhead efficiency.

For the purposes of comparison we have set the IFRC parameters \( r_{\text{init}} = 0.1 \text{ pkts/sec} \), \( \phi = 0.0125 \), \( \epsilon = 0.02 \). The upper queue threshold was set to 20 packets. These parameters were

Figure 5.10: Goodput and end-to-end packet delays for 20-node static case.
calculated as per the analysis presented in [67] for a 40 node topology, since this is the maximum size network we are dealing with in our experiments. For WRCP we set $\alpha = 0.1$, as per the analysis presented in section 5.3, and the empirical evidence presented in section 5.5.

### 5.6.2 Evaluation metrics and hypotheses

We list the key metrics considered in the experimental evaluation, along with hypotheses/expectations regarding WRCP’s performance on these metrics, given the design goals:

- **Goodput**: On these topologies with high quality links, we expect to see the goodput of sources match the allocated rates.

- **Flow Completion Time**: This refers to the time required to send a given number of packets end to end from a given source to the sink. As WRCP is designed for rapid convergence, we expect to see high performance with respect to this metric.

- **End-to-End Packet Delays**: Since WRCP uses an approximation of the achievable network capacity, it should be able to operate the system within the capacity region. A direct indication of this would be the ability of the protocol to maintain small queue backlogs. This in turn should lead to reduction of end-to-end delays due to queuing.

### 5.6.3 Comparative evaluation methodology

In order to have comparable results from WRCP and IFRC, we ran IFRC and WRCP over the same topologies (Figures 5.5 and 5.6).

Initially, we consider a scenario where all flows start at the same time, and all flows remain active for the entire duration of the experiment. We refer to this scenario as the static scenario. Since IFRC and WRCP both strive to achieve lexicographic max-min fairness, this scenario acts
as a validation for the WRCP design and implementation. This scenario also highlights the advantage of using rate control protocol (WRCP) that always makes the system operate with the rate region, in terms of the end-to-end packet delays.

We then consider dynamic scenarios where flows originate in the network at different times (hence the distinction with the static scenario). In these scenarios flows are intermittent. Certain flows remain on for the complete duration of the experiment, while a few flows turn on only for a portion of the experiment. The dynamic scenarios capture instances when short flows exist in the network. These scenarios will present the advantage that a explicit rate control protocol exhibits over an AIMD protocol in terms of higher goodputs, and hence better flow completion times. As mentioned earlier, it is important to note that faster flow completion times implicitly imply better energy utilization, since they result in shorter network uptime, conserving energy.

5.6.4 Static scenario

In these experiments, all nodes except the root node (node 12 for the 20-node topology, and node 29 for the 40-node topology) are sources. All flows start at the beginning of the experiment. Once a flow starts, it remains active for the duration of the experiment which lasted approximately 900 seconds (15 minutes).

Figures 5.9 and 5.11 presents the performance results of WRCP and IFRC on the 20 and 40-node topologies. For both topologies, it can be seen that the goodput of WRCP is better than or equal to that presented by IFRC (Figures 7.9(a) and 7.9(b)). Given that IFRC has the same objective as WRCP to present a lexicographic max-min fair vector, WRCP should present the same or a lexicographically greater vector than IFRC. Thus, these results highlight the functional correctness of WRCP. The gains of WRCP in this setting are reflected in the end-to-end packet delay.
performance of (figures 5.10(b) and 7.9(b)). Since WRCP uses explicit capacity information, it allows the system to operate within the rate region. IFRC, on the other hand, needs to constantly exceed the capacity region in order to estimate the capacity. The constant probing of IFRC results in the protocol exhibiting higher queue sizes than WRCP, resulting in larger end-to-end packet delays.

5.6.5 Dynamic scenario

In this section, we deal with a more practical setting where the work load, in terms of the number of flows active in the network, varies over time. This represents a more dynamic environment. In this setting, we show the gains of using an explicit capacity based rate control protocol over an AIMD based protocol. Specifically, we show that the gains are prominent when either the network dynamics are such that departure of flows will free a large amount of network capacity that can be consumed by the remaining flows, or the network experiences a large number of short flows. The gains are primarily in terms of shorter flow completion times, which will in turn
Figure 5.12: Goodput and delay performance for 40-node static case.
manifest themselves as energy savings. For each of the two topologies under consideration, we chose two different types of dynamic work loads to capture the different test scenarios. The two types of work loads, for each of the topologies is shown in figures 5.13 and 5.14. For each of the cases the x-axis of the figures represent the duration of the experiment, and the horizontal bars represent the duration for which each of the sources was sending traffic to the sink. To create a notion of a mix of long and short flows, long flows are represented by a subset of the flows that

Figure 5.13: Flow activity for the 20-node topology.
remain active for the entire duration of the experiment, while short flows are represented by the subset of flows that turn on and off intermittently during the experiment.

![Flow activity for the 40-node topology.](image)

**Figure 5.14:** Flow activity for the 40-node topology.

### 5.6.5.1 Case 1

For case 1, the rate allocation curves for both protocols, over both topologies, show that they adapt well to flow joins in the network. Both protocols cut down rate aggressively to avoid congestion collapse. The key difference in the protocol performance comes when flows depart the network.
If a large number of flows are active in the network, the per flow rate is quite small (1 pkt/sec for 19 active flows, and \( \sim 0.5 \) pkts/sec for 39 active flows). At this juncture if a majority of flows depart, suddenly a large amount of capacity becomes available for consumption. Such condition occurs at 2000 second for the 20-node topology, and at 2500 second for the 40-node topology.

The problem with IFRC under this condition is that since its rate of additive increase depends inversely on the \( r_{\text{thresh}} \) values, the rate increments are small, and its takes a long time for IFRC to consume this freed up capacity. WRCP, on the other hand, has explicit knowledge of the available capacity, it’s increments being independent of the current source rates and dependent purely on the available capacity. WRCP thus easily outperforms IFRC in consuming this freed up capacity. This is reflected in the goodputs for the flows that are active for the entire duration of the experiment in the 20, as well as the 40-node topologies (Figures 5.16 and 5.18).

A direct impact of the increase in goodput is a much smaller flow completion time. For the 20-node topology, the sources 7 and 13 are able to send out 9000 packets in 2600 seconds using WRCP, as compared to only 6000 packets under IFRC. For the 40-node topology, the sources
20 and 31 are able to send out 10000 packets in 3600 seconds for WRCP, as compared to only 7000 packets under IFRC. As mentioned earlier, shorter flow completion times will require short network up-time, and hence will result energy savings. The end-to-end packet delay performance for IFRC and WRCP (Figures 5.16 and 5.18) for this dynamic case also reflects on the ability of WRCP to operate within the capacity region.

5.6.5.2 Case 2

Unlike case 1, in case 2 the duration of the short flows is much shorter (\(~ 200\) secs). The gains of having an explicit capacity rate control protocol, for improving flow completion times of short flows, are clearly evident in this scenario. The goodput plots for both the 20-node and 40-node topologies (Figures 5.19 and 5.20). The goodput of the short flow for WRCP in both topologies is much higher than the goodput for short flows with IFRC. The long flows in WRCP get a lower goodput, since WRCP is more fair and allows a higher goodput for the short flows. IFRC, on the other hand, gives very high rates to long flows and starves the short flows. In short, WRCP gives a higher lexicographic max-min fair solution than IFRC. The increased goodput results in

Figure 5.16: Performance metrics for dynamic scenario (case 1) on the 20-node topology.
Figure 5.17: Rate allocation for dynamic scenario (case 1) on 40-node topology.

Figure 5.18: Performance metrics for dynamic scenario (case 1) on the 40-node topology.
Figure 5.19: Performance metrics for the dynamic scenario (case 2) on the 20-node topology.

very short flow completion times for the short flows. This is highlighted in terms of the packets delivered using WRCP in Figures 5.19 and 5.20.

Figure 5.20: Performance metrics for the dynamic scenario (case 2) on the 40-node topology.

To get a perspective on the gains exhibited by WRCP over IFRC in terms of the flow completion times, we can look at some of the number for packet delivery on the 20 and 40-node...
topologies. For the 20-node topology sources 2 and 16 are able deliver 450 packets in 200 seconds using WRCP, compared to only 50 packets using IFRC. For the 40-node topology, sources 17 and 33 are able to deliver \( \sim 500 \) packets in 200 seconds compared to 50 packets using IFRC. The delay performance of WRCP is far superior to IFRC for the 20-node as well as the 40-node topologies.

The two cases for the dynamic scenario exhibit the gains that WRCP presents for short flows as well as long flows in terms of flow completion times and delays.

5.6.6 WRCP performance in the presence of external interference

Since 802.15.4 experiences external interference due to 802.11 and other sources (such as microwaves), it is imperative that WRCP be able to perform efficiently in the presence of this external interference. In section 5.2.7, we described how WRCP uses the forwarding queues at a node to predict the amount of external interference. We validate this design choice in this section.

Figure 5.21(a) shows the rate allocation behavior for WRCP and IFRC on the 20-node topology in the presence of 802.11 interference. Recall that we are performing these experiments on channel 26 of 802.15.4. Only channel 14 of 802.11 can cause interference in channel 26 of 802.15.4. For these set of experiments we therefore operate an 802.11 radio, close to node 1, running the mad wi-fi driver on channel 14, transmitting UDP packets of size 890 bytes, in order to generate controlled interference. It is interesting to note that it is not only the power level of the interference, but also the rate at which this interference is generated that affects the achievable rates of a sensor network. This can be seen between 250-400 seconds, 550-700 seconds and 750-900 seconds. During these intervals the power of the external interference (802.11) was much
Figure 5.21: Rate allocation and queue length behavior with controlled external interference from an 802.11 source operating on channel 14 (2.482 MHz).

higher than the power at which the 802.15.4 radios. However the rate at which this external interference was being generated is was varied (17.8 KBps for 250-400 seconds, 890 KBps 550-700 seconds, 1780 KBps for 750-900 seconds). As can be seen in the rate allocation curve, at a lower rate (17.8 KBps) the external interference hardly affects the achievable rate of a sensor network, but as the rates start increasing the achievable rate starts going down, with the sensor network being completely shut down when the external interference starts operating at a high rate of 1780 KBps.
Figure 5.22: Goodput and End-to-end packet delay with external interference for 20-node topology.

Both IFRC and WRCP adapt their rates based on the amount of external interference. IFRC relies on coarse grained binary feedback asking nodes to simply cut their rates by half when it observes congestion. In the presence of external interference there is a high probability of these control signals being lost, resulting in nodes decrementing their rates by different amounts leading to lack of synchronization and unfair rate allocation between nodes. This can be seen in the rate allocation curve of Figure 5.21(b). The affect of this lack of synchronization can be seen
in the goodput (Figure 5.22(a)). WRCP therefore presents a lexicographically greater goodput than IFRC.

Figure 5.21(b) presents the instantaneous queue lengths, observed on nodes 7 and 13 (1-hop) for WRCP as well for IFRC. As can be seen, both protocols keep the queues bounded for the entire duration of the experiment. Due to the nature of algorithm used in IFRC (AIMD), it can also be seen that while for most of the experiment the sources are operating within the rate region the number of times sources cross the rate region, resulting in unsustainable queue sizes, is much higher in IFRC than in WRCP. This results in the delay performance of WRCP being much better than IFRC. This can be observed in Figure 5.22(b). Thus, WRCP operates efficiently in an external interference setting while still presenting performance improvements over the state of the art (IFRC).

5.7 Summary and Discussion

In this chapter we presented the Wireless Rate Control Protocol (WRCP), the first explicit and precise rate control protocol for a wireless sensor network. Through an extensive comparative evaluation with IFRC, an AIMD based rate control scheme, we highlighted the gains that can be achieved by designing a protocol based on explicit capacity information. Our comparative evaluation with IFRC shows that WRCP presents significant improvement in flow completion times and delay, especially in a dynamic flow scenario where there are a large number of short flows. The gains achieved in improving flow completion times indirectly manifests itself as energy savings as well, since with shorter flow completion times the active duration for radios can also be reduced, resulting in energy savings.
While the use of the *Receiver Capacity Model* (RCM) in designing WRCP is the key reason for the performance gains that WRCP presents, when compared to IFRC, the disadvantages of this design choice also needs to be stated. IFRC is agnostic to lower layer protocols due to its use of the AIMD mechanism for estimating congestion and avoiding it. This property allows IFRC to operate over any MAC protocol. WRCP, however, relies significantly on the receiver capacity information presented by RCM, which is currently very specific to the CC2420 CSMA MAC. Thus, a key drawback of WRCP as compared to IFRC is that it is tied to a specific MAC. In order to operate WRCP with another MAC, we will have re-calculate the receiver capacity values for the specific MAC protocol.
Chapter 6

Building a Flexible Rate Control Protocol

As seen in Chapter 5, the problem of rate control in wireless sensor networks has seen a rich set of proposals from a systems perspective ([20], [34], [63], [67], [71], [77], [93], [101]). Most of these protocols are designed purely from the perspective of congestion control and hence focus primarily on network stability. Many of these protocols implicitly or explicitly aim at providing some notion of fairness while achieving congestion control. However, to our knowledge, there is no prior work on practical rate control mechanisms for wireless sensor networks that provide the ability to optimize any generic concave rate-utility function.

In recent years, on the theoretical front, the literature on wireless networks has been enriched by several theoretical results that have developed new mathematical frameworks for design of optimal cross-layer protocols [14]. A particularly appealing stochastic network optimization approach that yields simple distributed algorithms for dynamic scenarios is based on the use of Lyapunov drifts [29].

The Lyapunov drift based techniques, described in [29], provide for distributed rate control based purely on local observations of neighborhood queues. They guarantee the stability of the

---

1This work was done in conjunction with Scott Moeller, and Prof. Bhaskar Krishnamachari, and was first published as [78].
system and also provide mechanisms to achieve optimization with respect to given utility functions. While attractive on theoretical grounds, to our knowledge these techniques have yet to be implemented in real wireless networks, only recently there have been attempts to transform this theory into practice ([97], [1]). One reason it has not been easy to translate the Lyapunov drift methodology from theory to practice is that it has been primarily developed under the assumption of a time-slotted system, implying a TDMA MAC. TDMA-based wireless networks are generally harder to implement due to the challenges associated with time-synchronization, particularly across multiple-hops. In this chapter, we first formulate the problem of rate control over a collection tree in wireless sensor networks, as a generic convex utility optimization problem. Using the Lyapunov drift framework, we then present a distributed algorithm and a proof of concept systems implementation that solves the convex optimization problem. Our primary contribution is to show that rate control algorithms based on the Lyapunov drift framework can be built over asynchronous CSMA-based MAC protocols. Specifically, for a wireless sensor network, we model the available bandwidth using the concept of receiver capacity (Chapter 4). This model introduces virtual queues in the Lyapunov drift framework that capture the interference constraints existing in the network. Our second contribution here is the experimental implementation of this distributed queue-based rate control algorithm on a real low-power wireless platform (the Tmote Sky from Moteiv) over a CSMA MAC (the CC2420 communication stack implemented in TinyOS 2.x). Our implementation results in the Backpressure-based rate control protocol (BRCP). The key novelty in BRCP is that, when using BRCP, the application developer is presented with a library of flow controllers; each flow controller can be designed to optimize a specific application defined concave rate-utility function. The library of flow controllers, thus,
give BRCP the flexibility to optimize for a wide range of concave rate-utility functions. BRCP therefore achieves our goal of designing a flexible rate control protocol.

6.1 Lyapunov optimization formulation

Our objective is to find a distributed algorithm to solve the following optimization problem:

\[
P_{\text{max}} : \sum g_i(r_i) \\
\text{s.t} \quad \vec{r} \in \Lambda
\]

Where \( g_i(r_i) \) is a concave rate-utility function of the allocated rate \( r_i \), and \( \Lambda \) is the capacity region of the network. For a wireless sensor network, the constraints of the problem \( P \), can be defined by the receiver capacity model presented in Chapter 4.

One approach to find a distributed solution to the problem \( P \), is to use the stochastic optimization framework proposed by Neely et al. [58]. The key attraction of using this specific framework, is the ability of algorithms resulting of this framework to arrive at a near optimal global solutions by making forwarding and flow control decision based purely on local queues. From an implementation perspective this property of the resulting algorithm minimizes the information exchange required for protocols designed based of them. This makes these techniques even more suited for wireless sensor networks due to the resulting savings in terms of energy.

The crux of this approach is to model the constraints in the optimization problem as virtual queues. For a given Lyapunov function, a bound on the the total drift of the system, which is a function of the virtual (formed of the constraints of the optimization problem) and physical queues (which are present in the forwarding engine of nodes) in the system, can then be found.
Minimization of the bound results in an algorithm, that makes transmission decisions based of the differential queue backlog of the node and its parent, and is guaranteed to stabilize the system. The transmission decision alone cannot find a solution to the problem $P$. In order to find a solution to the problem $P$, the Lyapunov drift based approach subtracts a reward function, based of the utility function $g_i(r_i)$, from the bound of the total system drift. Minimization of this bound then results in a flow control decision, along with the transmission decision formulated earlier. The flow control and the transmission decisions result in an algorithm that is guaranteed to stabilize the network and present near optimal solutions to the problem $P$.

The novelty in the analysis presented here, as compared to prior work such as ([56, 58]), is that the prior attempts at designing protocols using this technique assumed a detailed knowledge of the physical layer channel capacity. This was then used by a possibly centralized channel optimizer in order to ascertain optimal transmission rates per the Lyapunov drift minimization algorithm. The implicit assumption in the allocation is that the underlying MAC is TDMA. Though there is nothing in the analysis that limits the methodologies to a TDMA based approach. The Achilles’ heel of this approach seems to be that the optimization agent must have knowledge of the physical layer capacity region. The novelty of the solution presented here is that, using existing approaches, we show that the optimization problem can be applied to a CSMA based network as well. This is achieved by the additional constraints to the optimization problem which use the receiver capacity model.

We now present a Lyapunov drift based solution to the problem $P$. 
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i(t)$</td>
<td>The queue backlog for node $i$ at time slot $t$</td>
</tr>
<tr>
<td>$Z_i(t)$</td>
<td>Virtual queue backlog for node $i$’s collision domain at time slot $t$</td>
</tr>
<tr>
<td>$X_i(t)$</td>
<td>The attempted transmissions up the tree by node $i$ in time slot $t$</td>
</tr>
<tr>
<td>$\hat{X}_i(t)$</td>
<td>The actual transmission rate up the tree by node $i$ in time slot $t$</td>
</tr>
<tr>
<td>$R_i(t)$</td>
<td>The admitted exogenous arrivals to node $i$ in time slot $t$</td>
</tr>
<tr>
<td>$B_i$</td>
<td>The receiver capacity of node $i$</td>
</tr>
<tr>
<td>$D_i$</td>
<td>The collision domain of node $i$, includes neighbors and node $i$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>The set of one-hop children for which $i$ is the forwarding agent</td>
</tr>
<tr>
<td>$\hat{Z}_i(t)$</td>
<td>The sum of all virtual queues within $i$’s collision domain in time slot $t$</td>
</tr>
<tr>
<td>$V$</td>
<td>A tuning parameter that presents a tradeoff between the forwarding queue size in the system, and the rate-utility achieved.</td>
</tr>
</tbody>
</table>

Table 6.1: Variables used in the Lyapunov formulation.

### 6.1.1 Lyapunov optimization with receiver capacity virtual queues

Definitions of variables used in this Lyapunov formulation are given in Table 6.1. For our Lyapunov drift formulation we assume that the system operates on slot boundaries of duration $T$ seconds. For analytical tractability, we assume global synchronization between the nodes. We will relax this assumption when we describe our protocol implementation. Also, in a given slot $t$, the transmission rate of each node $i$, $X_i(t)$, is set to one of $\{0, B_{max}\}$. With $B_{max}$ as the receiver capacity. This way, nodes toggle between on and off modes of operation independently, with no concern for neighboring node’s activities. We rely on constraints provided by the receiver capacity model. The strength of our analysis lies in using the constraints provided by the receiver capacity model to characterize $\Lambda$ the capacity region of the network.

Using the Lyapunov drift approach, we first convert each of the constraints (provided by the receiver capacity model) in the problem $P$ to a virtual queue. Since a constraint is associated with each node $i$ (since every node in the network is a receiver), we associate a virtual queue $Z_i$ with each node.
The queuing dynamics for each of the *virtual* queues $Z_i(t)$ is given as follows:

\[
Z_i(t + 1) = \max[Z_i(t) - B_i, 0] + \sum_{j \in D_i} \hat{X}_j(t)
\]

(6.1)

Each time slot, the queue is first serviced (perhaps emptied), then arrivals are received. Each queue $Z_i$, therefore, first receives the sum of transmissions within the neighborhood of node $i$, and is then serviced by an amount equal to the receiver capacity of node $i$. Therefore, for every time slot in which neighborhood transmissions outstrip the receiver capacity of the node, this virtual queue will grow. In time slot $t$, $Z_i(t)$ thus represents the transmission volume by which the receiver capacity has been exceeded since $Z_i(t)$ last emptied. Every node also has a physical forwarding queue $U_i$.

The queuing dynamics of the *physical* queue $U_i(t)$ is similar to that of the virtual queues and is given by:

\[
U_i(t + 1) = \max[U_i(t) - X_i(t), 0] + \sum_{j \in C_i} \hat{X}_j(t) + R_i(t)
\]

(6.2)

That is, each node $i$ first attempts to transmit $X_i(t)$ units of data to its parent, then receives $\hat{X}_j(t)$ units of data from each child node $j$. Note that we differentiate here attempted transmissions ($X_i(t)$) and true transmissions ($\hat{X}_i(t)$). The difference being that while it may be most optimal to transmit a complete $X_i(t)$ units of data in this time slot, the queue may not contain sufficient data to operate optimally, so $\hat{X}_i(t) \leq X_i(t)$. 
Prior work, in the field of Lyapunov optimization (\cite{56, 58}), has shown that minimizing Lyapunov drift provides guaranteed stability over system inputs lying within the capacity region.

The Lyapunov drift of the above system of queues, $\Delta(\vec{U}(t), \vec{Z}(t))$, is defined as follows:

$$\Delta(\vec{U}(t), \vec{Z}(t)) = E[L(\vec{U}(t+1), \vec{Z}(t+1)) - L(\vec{U}(t), \vec{Z}(t))|\vec{U}(t), \vec{Z}(t)]$$

Where the Lyapunov function $L(\vec{U}(t), \vec{Z}(t)) = U^2(t) + Z^2(t)$. In order to calculate the above Lyapunov drift, we square the forwarding queue discrete time equation for the physical and virtual queues. Performing this operation on the discrete time equation of the physical queue yields the following:

$$U_i^2(t+1) - U_i^2(t) \leq -2 \cdot \left[U_i(t) \cdot \left\{X_i(t) - \sum_{j \in C_i} X_j(t) - R_i(t)\right\}\right]$$

$$+ [X_i(t)]^2 + \left[\sum_{j \in C_i} X_j(t) + R_i(t)\right]^2$$

(6.3)

Note that in typical systems, there exists a bound to the maximum value that $X_i(t)$ and $R_i(t)$ can achieve. Our constraints, in problem $P$, limited $X_i(t)$ to at most $B_{max}$. We also define a constant $R_{i,max}$, which is an upper bound on $R_i(t)$. We can then define the constant $G_i$ as follows:

$$G_i \equiv [B_{max}]^2 + \left[\sum_{j \in C_i} B_{max} + R_{i,max}\right]^2$$

$$\geq [X_i(t)]^2 + \left[\sum_{j \in C_i} X_j(t) + R_i(t)\right]^2$$

Similar calculations can be obtained for the virtual queues.
\[ Z_i^2(t+1) - Z_i^2(t) \leq -2 \cdot \left[ Z_i(t) \cdot \{ B_i - \sum_{j \in D_i} X_j(t) \} \right] + B_i^2 + \left[ \sum_{j \in D_i} X_j(t) \right]^2 \] (6.4)

Define constant \( K_i \) in a manner similar to \( G_i \):

\[
K_i \equiv B_i^2 + \left[ \sum_{j \in D_i} B_{max}(t) \right]^2 \\
\geq B_i^2 + \left[ \sum_{j \in D_i} X_j(t) \right]^2
\]

Substitution of \( G_i \) and \( K_i \) into equations (6.3) and (6.4), then summing over all nodes \( i \), and finally taking the expectation with respect to \((U_i(t), Z_i(t))\), yields the following Lyapunov drift bound:

\[
\Delta(\bar{U}(t), \bar{Z}(t)) \leq \Gamma
\] (6.5)

Where

\[
\Gamma = \sum_i K_i + \sum_i G_i - 2 \cdot \sum_i \left[ Z_i(t) E\{ B_i - \sum_{j \in D_i} X_j(t) | Z_i(t) \} \right] - 2 \cdot \sum_i \left[ U_i(t) \cdot E\{ X_i(t) - \sum_{j \in C_i} X_j(t) - R_i(t)|U_i(t) \} \right]
\]

As mentioned earlier, purely minimizing the bound on the drift given by equation (6.5) will result in a scheduling policy that will stabilize the system and give a bound on the average queue size within the system. However, by combining the global objective function of problem \( P \), with
the drift bound in equation (6.5) and minimizing this new bound, we can achieve a dual objective
of stabilizing the system and finding a solution to the problem $\mathbf{P}$.

Let $Y(t) = \sum_i g_i(R_i(t))$ be the system utility, as defined in the problem $\mathbf{P}$. We subtract $V \cdot E\{Y(t)|\vec{U}(t), \vec{Z}(t)\}$ from both sides of (6.5), yielding:

$$\Delta(\cdot) - V \cdot E\{Y(t)|\vec{U}(t), \vec{Z}(t)\} \leq \delta$$

Here $V$ is a constant parameter, and $E\{Y(t)|\vec{U}(t), \vec{Z}(t)\}$ is the conditional expectation of $Y(t)$. $\delta$ is given as follows:

$$\delta = \sum_i K_i + \sum_i G_i - 2 \sum_i Z_i(t)E\{B_i - \sum_{j \in D_i} X_j(t) | Z_i(t)\} - 2 \sum_i U_i(t) \cdot E\{X_i(t) - \sum_{j \in C_i} X_j(t) - R_i(t) | U_i(t)\} - V \cdot E\{Y(t)|\vec{U}(t), \vec{Z}(t)\}$$

In order to minimize the right hand side of equation (6.6), in expectation, it is sufficient to ensure we minimize the right hand side for every system state $(\vec{U}(t), \vec{Z}(t))$. We can neglect constant terms involving $K_i$ and $G_i$. The remaining terms can be separated into coefficients multiplying $X_i(t)$ and $R_i(t)$. The goal of our algorithm is then to minimize these terms through intelligent selection of per-time-slot decision variables $X_i(t)$ and $R_i(t)$. The resulting algorithm can be broken into two pieces: a transmission control decision and an admission control decision. A node performs a transmission control decision to determine whether it is optimal to forward packets up the collection tree. The admission control decision is performed in order to determine
if a local application layer packet should be admitted to the forwarding queue. We present the
details of the two control decisions below.

6.1.1.1 Transmission Control decision

Consider node \( i \) with parent node \( k \). The coefficient associated with transmission variable \( X_i(t) \)
is:

\[
- \left[ U_i(t) - \hat{Z}_i(t) - U_k(t) \right]
\]  
(6.7)

Where \( \hat{Z}_i(t) = \sum_{j \in D_i} Z_j(t) \) is the sum of the virtual queues of all nodes \( j \) in receiver
\( i \)'s neighborhood. Therefore, if transmission rates \( X_i(t) \) and \( X_j(t) \) are independent \( \forall i, j \), then
in order to minimize the right hand side of equation (6.6) we maximize \( X_i(t) \) \( \forall i \) such that equa-
tion (6.7) is negative. A node therefore transmits data to the parent whenever the differential
backlog between the node and its parent exceeds the sum of virtual queues within the local node’s
neighborhood.

6.1.1.2 Admission control decision

For node \( i \) admission rate \( R_i(t) \) is:

\[
- \left[ \frac{V}{2} \cdot g_i(R_i(t)) - U_i \cdot R_i(t) \right]
\]  
(6.8)

Therefore, in order to minimize the right hand side of equation (6.6) we maximize \( R_i(t) \) \( \forall i \) such
that equation (6.8) is negative. This equates to a simple admission control scheme. If the for-
warding queue size scaled by admission rate exceeds \( V/2 \) times the rate-utility for all admission
rates, then the admission request is rejected. Otherwise, a rate is chosen which maximizes equation (6.8).

Note that $V$, the tuning parameter that determines how closely we achieve optimal utility, appears only in the admission decisions. As $V$ grows, so does the queue backlog ($U_i(t)$) for which admissions are allowable.

An intuition for this behavior of $V$ can be obtained by looking at the feasible solutions of the optimization problem $P$. In the optimal solution of $P$, all the constraints in $P$ need to be tight. This implies that the system needs to be at the boundary of the capacity region, which further implies that system will be unstable (queue sizes will be unbounded). If we want to keep the system stable, we need to keep the constraints loose. This requires that the system to achieve a suboptimal solution with respect to the objective function, but ensures stability. Thus, $V$ tunes how closely the algorithm operates to the boundary of the capacity region.

### 6.2 Implementing Backpressure-based Rate Control using virtual queues

We will now show how the admission control decision and the transmission control decision can be used to implement a flexible rate control protocol, the Backpressure-based Rate Control Protocol (BRCP). Making BRCP operational in a CSMA-based network stack involves implementing the transmission control decisions which will inform the forwarding-engine when it is allowed to transfer packets to the CSMA MAC in order to forward it to the node’s parent. The admission control decision is implemented in the form of flow controllers that regulate traffic from the application layer into the system. The flexibility of BRCP results from the design of the flow
controller being based on the admission control decision. Since the admission control decision (equation (6.8) is dependent on the rate-utility function $g_i(r_i)$, by implementing a flow controller for each $g_i(r_i)$ we can present BRCP with a library of flow controllers that will give it the ability to optimize for any concave rate-utility function, making it a flexible rate control protocol.

Our target platform was the Tmote Sky class of devices. Tiny OS is the open source operating system that runs on the Tmote Sky and hence our software architecture was designed specifically to work with the TinyOS infrastructure.

Figure 6.1 presents the software architecture of our implementation. Because the objective was to perform distributed rate control over a static tree, we use the routing engine provided by the collection tree protocol in TinyOS-2.0.2 [26]. The routing engine helps build a tree rooted at a specific node, acting as the sink. We now present implementation details for each of blocks in Figure 6.1.
6.2.1 Hardware/Software implementation details

The rate controller block implements the transmission control decision given by equation (6.7). Though our analysis, presented in Section 6.1, assumes that all events occur on slotted time boundaries, this is not the case for a real CSMA based MAC. Hence the rate controller implements a timer that fires every $T$ seconds. The transmission control decision is thus made at the end of $T$ seconds. It’s essential to note that these timers are local to a node, making the decisions asynchronous. Note that in order to carry out the transmission control decision based on equation (6.7), knowledge is needed of the number of transmissions ($X_i(t)$) and virtual queue sizes ($Z_i(t)$) of all neighbors during the prior time slot. Therefore, once every time slot, each node attaches their transmission rate and virtual queue size to the next transmitted MAC header and broadcasts the data packet instead of unicast it. This allows all neighboring nodes to overhear this information and retrieve it from the headers.

To aid in the transmission control decision, the rate controller block is responsible for estimating the node’s current transmission rate by maintaining an exponential weighted moving average of the number of packets transmitted by the forwarding engine, on a per slot basis. In addition to rate estimation, the rate controller block updates the local virtual queue using equation (6.1). This requires knowledge of the local node’s receiver capacity ($B_{max}$), its current transmission rate ($X_i(t)$) and the transmission rate of all its neighbors who are active in its broadcast domain. The rate controller establishes its receiver capacity by setting it to the saturation throughput corresponding to the number of neighbors within its broadcast domain (Chapter 4). In this chapter, we have hard-coded the receiver capacities to 70 packets per second, a safe lower bound to the
optimal capacity; the packet size is 40 bytes. Techniques could be implemented that would improve system performance by estimating the number of active neighborhood transmitters in each time slot. We feel that for our basic proof of concept, such techniques lie outside the scope of this work. The rate controller block informs the forwarding-engine, as to when it is allowed to transfer packets to the CSMA MAC, in order to transmit the packets to the node’s parent.

The flow controller block implements the admission control decisions. Although the analysis presented in section 6.1 holds for any concave utility function $g_i$, we restrict our implementation to three types of flow controller that try to achieve maximization of network efficiency, proportional fairness and $\alpha$-fairness. We describe the details of each of the flow controllers below:

- **Linear Utility Flow Controller (Maximizing network efficiency):** By assuming linear $g(R_i(t))$ we can simplify our admission decisions. Let $g(R_i(t)) = u_i \cdot R_i(t)$. Note that this reduces our admission decision components of the Lyapunov drift bound to the following:

$$-R_i(t) \cdot \left[ \frac{V}{2} \cdot u_i - U_i \right]$$

(6.9)

Under linear utility functions, the admission rate $R_i(t)$ becomes inconsequential, allowing for the following simple admission criterion: for node $i$, if $\left[ \frac{V}{2} \cdot u_i - U_i(t) \right] > 0$ then admit the application-layer send request.

- **Proportional Flow Controller:** For a proportional flow controller $g(R_i(t)) = \log(R_i(t))$.

Hence the admission decision is to set the rate $R_i(t)$ so as to maximize the following equation:

$$\max : \frac{V}{2} \log(R_i(t)) - U_i(t)R_i(t)$$

(6.10)
Taking the first derivative of the above solution results in a simple admission decision: In order to achieve proportional fairness set the admission rate $R_i(t) = \frac{V}{2U_i(t)}$.

- **α-fair Flow Controller:** Taking a similar approach to the proportional fair controller, it can be seen that the admission decision for the α-fair controller is to set the admission rate
  \[ R_i(t) = \alpha \sqrt{\frac{V}{2U_i(t)}}. \]

As can be seen, a flow controller is associated with a specific objective function $g_i(r_i)$. Thus by providing a library of flow controllers (as shown above), we can make BRCP a flexible rate control protocol.

The forwarding-engine maintains the forwarding FIFO queue. This queue receives packets both from local admissions and from children in the tree structure. As mentioned in Section 6.1.1, our analysis assumes that nodes transmission rate $X_i(t)$ can take one of two values \( \{0, B_{\text{max}}\} \); our implementation therefore allocates transmission tokens to the forwarding-engine in an all-or-none manner. At each time slot, a node, based on its transmission control decision will either allocate $B_{\text{max}}$ or zeros forwarding tokens for the forwarding-engine. This is where the virtual queues become critical, as they ensure that our time average receiver bandwidth constraints are not violated, even while individual nodes attempt to transmit at maximum rate during their active time slots.

The communication stack consists of the default TinyOS CSMA mechanism for the CC2420 radios, which is the 802.15.4 radio present on the Tmote sky platforms.
6.2.2 An illustrative example

We now provide a simple illustrative example to give some intuition on the admission and transmission control decisions. Consider a five node linear forwarding topology as in topology 1 of Figure 6.3. All nodes are considered to be traffic sources (omitting the sink, node 1). All nodes are assumed to be within range of one another, implying that interference links exist between all nodes. Assume a linear $g(R_i(t)) = u_i \cdot R_i(t)$. Let $[u_5, u_4, u_3, u_2, u_1] = [3, 6, 2.5, 1, 0]$, $V = 20$, and $B_{max} = 1$. For this example, we assume that we have infinite sink-destined data at each of the sources.

With the linear topology and linear utilities defined here, one can quickly determine the optimal rate allocation scheme. Node 5 provides three utility while requiring four transmissions per unit of data that reaches the sink. This gives it an efficiency of $\frac{3}{4}$. Repeating this computation yields efficiencies $[e_5, e_4, e_3, e_2, e_1] = [\frac{3}{4}, 2, 1.25, 1, 0]$. Therefore, per unit of data transmitted,
node 4 provides the highest system utility. Therefore, in order to maximize the global rate-utility 
\[ \sum_i (g_i(r_i)) \], the entire system capacity should be allocated to node 4.

In Figure 6.2 we depict convergence towards optimality. Figure 6.2(a) depicts sub-optimal 
operation in the startup phase, a result of sub-optimal virtual queue sizes. In this phase, traffic 
sources at nodes 4, 3 and 2 are able to forward data to the sink. In Figure 6.2(b), the virtual 
queues grow, creating back-pressure which obstructs node 2’s local admissions. In this phase, the 
back-pressure caused by neighborhood virtual queues is still insufficient to halt local admissions 
by node 3. Therefore, traffic continues to flow from sources at nodes 4 and 3. Finally, in Figure 
6.2(c), the virtual queues grow to their equilibrium levels. At this point, the per-hop back-pressure 
is sufficient to halt localized admissions by both of nodes 2 and 3. Only node 4 has a sufficient 
forwarding queue backlog necessary to overcome the back-pressure. We describe the three phases 
in greater detail below.

Recall from equation (6.9) that if the queue backlog is greater than \( \frac{V}{2} \cdot u_i \), we reject the applica-
tion layer admission request. This implies that local admissions will halt above pre-determined 
queue backlog thresholds, equal to \( \lfloor \frac{V}{2} \cdot u_i \rfloor \). For the current topology, queue thresholds for 
\([U_5, U_4, U_3, U_2]\) are therefore \([30, 60, 25, 10]\).

Figure 6.2(a) depicts the early startup phase. In this phase, the virtual queues are small 
because the nodes have only recently been in violation of the time average receiver capacity 
constraint. Because the virtual queues are small, condition (6.7) dictates that in order for nodes to 
transmit, the forwarding queues need not exceed the local admission thresholds. As the admission 
threshold is not being exceeded, we will see traffic from sources at nodes 3 and 2 entering the 
system. This is a suboptimal behavior.
Figure 6.2(b) depicts the system as it moves towards equilibrium. As transmission rates continue to violate receiver capacity, the virtual queues grow. This backpressure causes the forwarding queues to sequentially grow above their local admission thresholds. In this figure, for example, the value of $Z_2(t)$ has grown to 12. A forwarding queue backlog of at least 12 packets is therefore required in order to transmit to the sink. This exceeds node 2’s admission threshold of 10 packets. Therefore all future local admission request will be rejected by node 2.

Finally, Figure 6.2(c) depicts the system at equilibrium. Here, a special condition holds. The sum of $Z_i(t)$ between node 4 and the sink is exactly equal to node 4’s admission threshold of 60 packets. No other node in the network is allowed to admit local traffic, as the virtual queues have grown too large for any non-optimal source. At this point node 4 is the only active source in the network.

6.3 Experimental evaluation

In order to verify proper operation of BRCP, we performed experiments using the Tmote Sky class devices running TinyOS-2.0.2. We considered three test scenarios. Two five node topologies were selected, as highlighted in Figure 6.3. Table 6.2 indicates the utility parameters and topology for each of the four test scenarios. Table 6.3 documents the parameters common to all scenarios.

These configurations were loaded onto five Tmote Sky devices. In experimentation, each scenario was allowed to run for 25 minutes. Each node recorded the forwarding and virtual queue backlogs, while the root node recorded goodputs received from nodes in the system. In all tests, the networks were fully connected.
Using the receiver capacity model, based on linear utilities and topologies we can pre-compute the optimal rate allocation for each of the scenarios. Optimal per-node transmission rates for all scenarios are given in Table 6.4.

### 6.3.1 Experimental results

For each of the scenarios, the results of the four 25-minute experiments are plotted in Figures 6.4, 6.5 and 6.6. Time average goodputs have been recorded in Table 6.5.
Figure 6.4: Plots of results for scenario 0.

(a) Average goodput rates received at the sink.

(b) Local forwarding queue sizes.

(c) Local virtual queue backlogs.
Figure 6.5: Plots of results for scenario 1.

(a) Average goodput rates received at the sink.

(b) Local forwarding queue sizes.

(c) Local virtual queue backlogs.
Figure 6.6: Plots of results for scenario 2.
Table 6.4: Optimal packet rate allocations subject to receiver bandwidth constraints for scenarios 0 through 2.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>node5 PPS</th>
<th>node4 PPS</th>
<th>node3 PPS</th>
<th>node2 PPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>23.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

6.3.1.1 Optimality gap

Comparison with the optimal solutions recorded in Table 6.4 indicates that there is a consistent gap between achieved experimental results and analytically calculated optimal transmission rates.

Two possible explanations for this gap are as follows. First, there is some overhead required in our system implementation. As the root node never generates or forwards traffic, there would be no opportunity to broadcast virtual queue backlogs. We therefore require a broadcast packet be sent by the root node every second. This packet is then accounted for by the system and cuts into the receiver capacities of all nodes within the root's neighborhood. In our test environment, all nodes overhear the root, so all nodes incur costs within their virtual queues as a result of this mandatory broadcast. Additionally, consider that leaf nodes often are not optimal, and that nodes sitting behind the optimal traffic generator are very unlikely to ever forward or generate packets once equilibrium is reached. The effect of this, however, is that their broadcast updates containing real and virtual queue backlogs will halt, resulting in stale system state information. We therefore also require that all nodes not admitting or forwarding any traffic must still generate a minimum of one packet per second. The optimality gap is therefore variable and depends upon the number of inactive nodes.

Second, there is an optimality gap between the actual optimal and Lyapunov formulations [29], and it is a function of $V$. This suboptimal behavior is clearly visible in scenario 1. Note that in
Figure 6.5(a), node 2 receives 4.28 packets per second, though node 2 is not in the optimal solution. The optimal source node for this scenario is node 3 with $w_3 = 3$. Based on the control decision in equation (6.9), with $V = 20$, node 3 therefore admits packets up to a forwarding queue size of 30 packets. At equilibrium, as node 3 is two hops from the sink this leads to a neighborhood virtual queue backlog equal to 15, which is evident in Figure 6.5(c). Since the forwarding queue backlog at node 2 is no more than 15 and given our rather large per time slot token allocations relative to these queue sizes, occasionally the queue of node 2 empties below 10. This is the admission threshold for node 2 and hence when forwarding queue size goes below this threshold, source traffic from node 2 is admitted. This leads to the suboptimal behavior observed in Figure 6.5(a). To make the rate allocation closer to optimal, we would need to enlarge our value of $V$ such that the differential in forwarding queues between node 2 and node 3 would be sufficiently large. Increasing $V$ was not an option in our setup, as the queues had already reached maximum sustainable values on our Tmote Sky device (50 packets).

6.3.1.2 Virtual queue behavior

From a theoretical standpoint, in a fully connected topology, all local virtual queues should have identical backlogs. However, in our experiments, note that all local virtual queues exhibit slow drifts away from their theoretical values. The drift results from inaccurate updates of local virtual queues using stale neighbor transmission rate information. The stale neighbor transmission rate information occurs due to the asynchronous nature of the system. Despite the existence of this drift, the system performance is not affected because the performance does not depend on the individual virtual queue sizes but the sum of the virtual queues in the neighborhood. As the graphs reflect the sum of the virtual queues remains constant, thus no adverse effect on rate allocation.
is observed. Verifying proper behavior under virtual queue drift for systems with dynamic traffic flows is a topic for future investigation.

### 6.4 Implementing BRCP without virtual queues

BRCP presented in this chapter has two components: the flow controller, which does not re-
quire information about the virtual queues, and the backpressure scheduler (implemented in the form of the rate controller block in Figure 6.1), which is based on the receiver capacity model and hence relies on the virtual queue information to make its transmission decisions. Theoret-
ically, the backpressure scheduler has two roles: first it is supposed to improve the rate region of the underlying MAC; second it uses the backpressure mechanism to convert queue sizes into cost information, that can than be used by higher layer protocols to implement routing and rate control functionality (in this chapter we have focused purely on the rate control functionality). Since the receiver capacity model is an approximation of the true rate region, the use of virtual queues will force the backpressure scheduler to perform within the capacity region of the existing CSMA MAC. Thus, the backpressure scheduler designed using the virtual queue information cannot improve the capacity region of the existing CSMA MAC. The use of virtual queues for designing the backpressure scheduler is therefore restrictive in some sense. It is therefore neces-
sary to decouple the flow controller component from the backpressure scheduler, and see if we can get better performance from the stack by running the flow controller over heuristics that try

<table>
<thead>
<tr>
<th>Scenario</th>
<th>node5 PPS</th>
<th>node4 PPS</th>
<th>node3 PPS</th>
<th>node2 PPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.1</td>
<td>19.3</td>
<td>0.0</td>
<td>1.8</td>
</tr>
<tr>
<td>1</td>
<td>1.1</td>
<td>0.8</td>
<td>26.5</td>
<td>4.3</td>
</tr>
<tr>
<td>2</td>
<td>30.0</td>
<td>1.1</td>
<td>0.7</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Table 6.5: Achieved packet rate allocations for experimental scenarios 0 through 2.
to emulate backpressure scheduling without the use of virtual queue information. Heuristics that do not rely on virtual queue information might present a better capacity region than the approximation presented by the receiver capacity model, resulting in an improved performance for the backpressure-based rate control protocol.

To test the above hypothesis, we compare the analytically obtained value for the flow controllers using the receiver capacity model, and compare it against those obtained using experiments run using two different heuristics for the backpressure scheduler: the “maximum differential queue (MDQ)” scheduler and the “positive differential queue (PDQ)” scheduler. We first describe the two schedulers before presenting our results.

6.4.1 Backpressure with differential queue prioritization (MDQ)

The MDQ scheme sits on top of an existing CSMA MAC. In MDQ, every node transmits its queue differential as part of the header of each packet transmission. Nodes in a given neighborhood, store the queue differential of all nodes within neighborhood and compare their own queue differential with the maximum differential observed amongst these stored values. A node then transmits only if it has the maximum differential in its neighborhood. This scheme does not require any modification to the MAC layer except an extra eight bits in the header for the queue differential information. The MDQ scheme is similar to the one proposed by Akyol et al. [1]

The flow controllers presented in section 6.2 can now work on top of the new proposed scheme without any modifications, since their decisions are still based of the queue backlog independent of the control decision. We have implemented the MDQ mechanism on TinyOS-2.0.2.2 for the Tmote sky devices following the software architecture presented in figure 6.1. We have implemented the MQD mechanism as part of the rate controller block.
6.4.2 A pure back pressure implementation (PDQ)

A simple implementation of the back pressure based rate control protocol (which we term as positive queue differential (PDQ)), is to make control decisions purely of the queue backlog differential, without regard to the differential in the neighborhood. In this mechanism the forwarding engine injects a packet into the MAC, as long as the node has a positive queue differential; irrespective of the size of the queue differential. We choose this mechanism since the Lyapunov drift analysis suggests such an approach for wired networks, and hence the need to verify this mechanism in a wireless setting.

6.5 Empirical evaluation of BRCP without virtual queues

In order to evaluate the above scheme, which implements backpressure based rate control without the use of virtual queues, we ran the MQD and PDQ scheme on the 20-node multi-hop topology shown in figure 6.7. The parameters used for the experiments were as follows:
• For the MDQ scheme we set \( V=100 \), while using the proportional fair controller, and \( V \) was set to 60000 and \( \alpha = 7 \) while using the \( \alpha \)-fair controller. For our experiments, since the objective of using the \( \alpha \)-fair controller was to achieve a max-min fair solution, we set \( \alpha \) to a high enough value.

• For the PDQ scheme, we set \( V=100 \) while using the proportional fair controller. \( V \) was set to 30000 and \( \alpha = 7 \) while using the \( \alpha \)-fair controller.

Figure 6.8: Goodput for the MDQ scheme, using a proportional fair controller and an \( \alpha \)-fair controller.
Table 6.6: The sum-log utilities achieved for the multi-hop topology using the proportional fair controller with the PDQ and the MDQ schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Proportional Fair utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDQ</td>
<td>22.56</td>
</tr>
<tr>
<td>PDQ</td>
<td>29.95</td>
</tr>
<tr>
<td>Analytical</td>
<td>21.05</td>
</tr>
</tbody>
</table>

Figures 6.8(a) and 6.8(b) show the goodput achieved, by a proportionally fair controller and an $\alpha$-fair controller, running over an MDQ scheme. Figures 6.9(a) and 6.9(b) show the goodput achieved by a proportionally fair controller and an $\alpha$-fair controller, running over a PDQ scheme.

For the proportional fair controller, with either scheme it can seen that some nodes get a much higher utility than the analytically predicted values. This reflects in the sum utility when considering the proportionally fair controller. Table 6.6 presents the sum-log utility using either scheme with the proportional fair controller. It is evident that either of the heuristics presents a better utility than the analytically predicted values.

For the $\alpha$-fair controller the goodput of the PDQ and the MDQ scheme is comparable (or slightly) better than the analytically predicted values. Thus, these results suggest that it is better to use the backpressure stack with the flow controllers designed using the Lyapunov drift analysis presented in this chapter, over backpressure scheduling heuristics that are not constrained by the underestimation of the rate region presented by the receiver capacity model. The question that remains to be answered is, which heuristic is best suited to implement the backpressure based scheduler at the MAC layer. In chapter 7, we explore the entire design space of a backpressure-based stack and answer the above question, while at the same time exploring the parametric dependance of the rate-utility performance of the stack on the parameter $V$.
Figure 6.9: Goodput for the PDQ scheme, using a proportional fair controller and an $\alpha$-fair controller.
Chapter 7

Exploring the Design Space of a Backpressure-based Protocol

7.1 Introduction

In chapter 6, we designed a Backpressure-based Rate Control Protocol (BRCP), using Lyapunov drift analysis based on the receiver capacity model. We also show that the flow controllers of BRCP, obtained from the Lyapunov drift analysis and presented in chapter 6, are independent of the virtual queues (obtained from the receiver capacity model) and hence can be used in an implementation of BRCP that does not require receiver capacity information. Further, in chapter 6, Section 6.4, we show that such an implementation will actually present a better performance than a rate control protocol that uses a backpressure-based scheduler designed using receiver capacity virtual queue information. Therefore a more practical backpressure-based rate control protocol design will involve the implementation of backpressure-based scheduling heuristics at the MAC layer (independent of the receiver capacity model), and implementation of the flow controllers described in chapter 6 at the transport layer.

1This work was done in conjunction with Scott Moeller, and Prof. Bhaskar Krishnamachari, and was first published as [80]. A more detailed technical report on this work is available as [81].
Given such a protocol stack, in this chapter we evaluate the design choices at the MAC and transport layer, that are necessary to implement backpressure-based protocol in a wireless sensor network\(^2\). We perform this evaluation by implementing a backpressure-based rate control stack, as part of the Tiny OS-2.x communication framework, and evaluating the stack over the USC Tutornet testbed [46]. This is a 94 node wireless sensor network (WSN) spanning two floors. The testbed is composed of Tmote sky devices [55], that are capable of running the Tiny OS-2.x operating system. To the best of our knowledge, this is one of the first evaluation’s of a backpressure-based protocol stack performed in the context of a WSN. Our evaluation helps us garner insights into design issues related to MAC layer modifications, and transport layer parameter settings, helping us understand the design choices required to simplify backpressure rate control design, and improve rate-utility performance of these protocols in a practical sensor network setting.

Our evaluation aims to ask two key questions pertaining to the MAC and the transport layer. At the MAC layer, existing heuristics ([1], [96]) try to emulate the optimal backpressure scheduling policy [89] by tying the back-off window size of the CSMA MAC to the queue differential of a node with its next-hop. By linking the back-off window size to the queue differential, the proposals try to present nodes with larger queue differentials a higher probability of transmission. These modifications hope to achieve two objectives; First, since they emulate the optimal backpressure scheduling policy they should be able to improve the rate region of the original CSMA MAC. Second, the backpressure mechanism makes the queues of the nodes represent the amount of congestion a node is causing while operating at its current source rate. Since there is no theoretical justification to these MAC layer modifications, it is unclear if the complicated queue

\(^2\)Note, though backpressure-based techniques allow for routing decisions as well, we focus our efforts on MAC and transport layer issues assuming shortest path routing.
prioritization techniques proposed actually improve the existing rate region. Without improvements to the rate region, the use of queue prioritization techniques, and hence modifications to the MAC layer become redundant. Since, the second objective (making queues represent congestion cost) can be achieved using a much simpler mechanism which allows forwarding by a node whenever a node has a positive queue differential, requiring no modification to existing CSMA MAC. Thus, we raise the following question: *Is queue prioritization essential for implementing backpressure-based protocols over a CSMA based MAC?*

At the transport layer, backpressure-based protocols require a new type of flow controller that maps the instantaneous queue size to an instantaneous source rate. The mapping functions are derived using the theoretical frameworks proposed by either Stolyar [83] or Neely *et al* [58], and are of the form \( r_i(t) = f(V, U_i(t)) \). Here \( r_i(t) \) is the instantaneous source rate of node \( i \), \( V \) is a constant parameter\(^3\), and \( U_i(t) \) is the instantaneous queue size. The constant parameter \( V \) presents a tradeoff between queue size and rate-utility performance ([1], [58]). For these protocols, rate-utility performance improves logarithmically with increase in queue size [58]. Given finite queues, and drastic reduction in per flow rate (due to increased interference) as flow count increases, it remains unclear whether good performance can be achieved under fixed parameter settings. Thus, the question we raise pertaining to this parameter \( V \) is: *Is there a single value of \( V \), for a given topology, that will allow backpressure-based rate control to perform competitively to existing rate control protocols, irrespective of the number of active flows in the system?*

Through the evaluation presented in this chapter, we obtain the following three findings:

---

\(^3\)The notation \( V \) for this parameter has been borrowed from the work by Neely *et al* [58], the work by Stolyar [83] introduces a similar parameter called \( \beta \).
First, a comparative evaluation of different heuristics for backpressure scheduling over a CSMA MAC shows that in a sensor network setting, backpressure-based rate control protocol can be implemented without any MAC layer modifications. This can be achieved by implementing a simple scheme at the network layer that injects packet into the MAC only if a node has a positive queue differential.

Second, a comparative evaluation with an existing rate control protocol (IFRC [67]) demonstrates that for a given topology there exists no single value for the parameter $V$, that can guarantee optimal performance to backpressure-based rate control. We further show that the optimal parameter setting is a function of the number of flows active in the network, and automatic parameter adaptation is required for backpressure-based rate control protocol to perform well in a dynamic flow scenario.

Third, our evaluation also shows that with optimal parameter settings (adapting $V$ to the number of active flows) backpressure-based rate control protocol can easily outperform existing rate control stacks, by a factor of about 2.

This chapter is organized as follows: in Section 7.2, we present a software architecture that captures the general design of a backpressure-based rate control stack. In Section 7.3, we present the implementation details of heuristics that have been proposed for implementing backpressure scheduling over a CSMA stack. In Section 7.4, we present a comparative empirical evaluation, of the different heuristics that can be used for implementing backpressure scheduling in a CSMA based wireless network. In Section 7.5, we present an evaluation of the backpressure based rate control protocols against IFRC [67] in order to understand the parameter dependence of backpressure protocols.
7.2 Backpressure-based rate control stack

We start our investigation by presenting a software architecture (Figure 7.1) that captures the design of a generic backpressure-based rate control stack.

![Figure 7.1: The software architecture of a backpressure based rate control stack.](image)

We restrict our investigation specifically to a fixed collection tree, implying that there exists a single destination in the network to which all sources are routing their data.

When routing is fixed, backpressure-based rate control stack is implemented at the MAC and the transport layers. The transport layer functionality is implemented as part of the “Leaky Bucket” and “Flow Controller” blocks in Figure 7.1. The flow controller needs to determine the allowed instantaneous rate of admission as a function of the forwarding queue size. The “Flow Controller” block in Figure 7.1 interacts with the forwarding-engine to learn the instantaneous queue size, and sets an allowed admission rate in the leaky bucket. The leaky bucket then generates tokens at the admission rate. When a packet arrives from the application to the flow controller, it is injected into the forwarding-engine only if a token is available.
The backpressure-based MAC is implemented as part of the “Forwarding Engine” and “Communication stack” blocks (Figure 7.1). The forwarding-engine calculates the current queue differential, using information about parent queue size (learned through periodic broadcasts) and its own queue size. Based on the current queue differential, the forwarding-engine decides whether or not to transfer a packet to the MAC layer (represented by the communication stack in figure 7.1). If the scheduler wants to implement differential queue prioritization, the forwarding-engine can use interfaces provided by the underlying MAC to modify the MAC back-off window sizes, before injecting the packet.

The general architecture of the backpressure-based rate control stack, captures the underlying software design for the various versions of the backpressure-based rate control protocol (BRCP) (Chapter 6) implementation, that can be obtained by changing the implementation of the backpressure scheduler at the MAC layer. We now describe the implementation of the transport and MAC layer in further detail.

### 7.2.1 Transport layer

The key component in implementing the transport layer is the flow controller block. The objective of the flow controller is to operate the source at a time average rate \( r_i \), allowing the source to achieve a utility \( g_i(r_i) \), such that the rate allocation \( r_i, \forall i \), maximizes \( \sum_i g_i(r_i) \) across the entire network. \( g_i(r_i) \) is assumed to be a concave utility function. Note that the flow controller runs at each node, and hence it needs to make local decisions, but the local decisions should be such as to optimize a global function \( \max \left( \sum_{\forall i} g_i(r_i) \right) \). In order to design such flow controllers we can use one of two techniques presented by Stolyar [83] and Neely et al. [58].
In the Chapter 6, the flow controller is designed using a technique proposed by Neely et al. [58]. In this design, at every time step $t$, the instantaneous source rate $R_i(t)$, at which packets are admitted into the system is that which maximizes the following equation:

$$\max \left[ \frac{V}{2} \cdot g_i(R_i(t)) - U_i(t) \cdot R_i(t) \right]$$

(7.1)

This results in a simple solution. Set $R_i(t)$ to a value that satisfies the following equation:

$$\frac{V}{2} \cdot g_i'(R_i(t)) = U_i(t)$$

(7.2)

Here $V$ is a constant that acts as a tuning parameter to effect a tradeoff between the forwarding queue size $U_i$, and utility $g_i(r_i)$. A large value of $V$ will imply large value of $U_i$, and large $g_i(r_i)$. Whereas a small value of $V$ will imply small value of $U_i$, and small $g_i(r_i)$.

### 7.2.2 Backpressure-based MAC

![Diagram of backpressure scheduling](image)

Figure 7.2: Understanding backpressure scheduling.

---

4It should be noted that flow controllers in the proposal by Akyol et al. [1], are very similar in structure to the one shown in equation 7.2. The only difference between the two designs is the parameter $\beta$, which has an inverse effect as compared to $V$. 

137
Ideally a backpressure-based MAC should implement the scheduling policy proposed by Tassiulas et al. [89]. Figure 7.2 shows a fully connected single hop wireless network. Nodes 2, 3, and 4 are sources, and node 1 is the sink. The queue differential between a source $i$ and node 1, at time $t$, is given by $U_i(t) - U_1(t)$. In the optimal backpressure-based scheduler, if nodes are contending to transmit to the sink, and link rates for all sources is assumed equal, the backpressure scheduler will select the node with the largest queue differential. The optimal backpressure scheduler assumes a TDMA MAC, which will present it with a maximum match schedule, which is NP-hard to implement.

The challenge in implementing such a scheduling policy in a CSMA based system is that a CSMA MAC makes purely distributed decisions, with the only mechanism of controlling when a node accesses the channel being the size of the back-off window. Proposals ([1, 96]) therefore try to achieve prioritization of node transmission by changing the CSMA window size based on the current queue differential. We refer to these techniques collectively as queue differential prioritization based techniques. Akyol et al. [1] achieves queue differential prioritization by making nodes choose one of two window sizes. Nodes having the highest weight in a neighborhood choose the larger window size, and all other nodes choose a smaller window size. The weight of a node is the product of its queue differential and its current link rate. We refer to the heuristic proposed by Akyol et al. [1] as the Max Differential Queue MAC (MDQ).

Warrier et al. [96] achieves queue prioritization by having queue differential thresholds mapped to corresponding window sizes. When a nodes queue differential falls between two thresholds it chooses the corresponding window size. In this scheme, the larger thresholds are mapped to smaller window sizes and smaller thresholds to larger window sizes. We refer to this particular scheme as the Differential Queue Window Mapping MAC (DQWM).
Given the sub-optimality of the proposed heuristics, it is worthwhile considering a much simpler approach that does not strive to change the existing rate region, but still results in a backpressure signal to the source. In this scheme the forwarding-engine is allowed to transfer packets to the MAC only if a node has a positive queue differential with its parent, irrespective of the size of the differential. We refer to this scheme as the *Positive Differential Queue MAC* (PDQ). Note that in this scheme once the packet reaches the MAC, the MAC uses the default widow sizes to attempt transmission. This scheme is similar to the one used by Radunovic *et al* in [65].

Since our goal is to ascertain the necessity of queue differential prioritization techniques, in the next section we present the implementation details of these schemes over an existing CSMA MAC.

### 7.3 Enabling backpressure over CSMA

The target platform for presenting our evaluation of backpressure based protocols in WSN is the Tmote Sky device [55]. The Tmote Sky platforms communicate using IEEE 802.15.4 compatible CC2420 radios, and can run TinyOS-2.x. This OS has a CSMA MAC for the CC2420 radios.

In this section we present the implementation details of the differential queue prioritization heuristic proposed in [1, 96] (MDQ, DQWM), and PDQ over the CC2420 CSMA MAC. We first present a description of the CC2420 CSMA MAC over which the schemes will be implemented.
7.3.1 The CC2420 CSMA MAC

The CSMA-CA algorithm in CC2420 CSMA MAC [90] uses two types of back-off windows, the initial back-off window \( W_i \), and a congestion back-off window \( W_c \). When a node injects a packet into the MAC layer, the MAC layer performs a random back-off between \([0, W_i]\) at the end of the initial back-off phase the MAC performs a carrier sense to determine if the channel is free. On finding the channel free it transmits the packet. However, if the channel is busy, the MAC enters a congestion back-off stage performing a random back-off between \([0, W_c]\). When the congestion timer expires, the MAC repeats the carrier sense process. Retransmissions are implemented as part of the MAC to provide link layer reliability. The default value for \( W_i = 320 \), and default value for \( W_c = 80 \). The back-off slot duration is 32.25 microsecond resulting in a 10 ms \((320 \times 32.25)\) initial back-off window, and 2.58 ms \((80 \times 32.25)\) congestion back-off window.

7.3.2 Implementing MDQ MAC

The maximum differential queue prioritization technique (MDQ), proposed by Akyol et al. [1], and described in section 7.2.2, was implemented over the CC2420 CSMA as follows. Two fields were added to the CSMA header: a field to contain the current queue size of the node and a field to contain the current weight of the node. A node informs its neighbors of its weight and its current queue size using the extra fields in the CSMA header, periodically broadcasting data packets instead of uni-casting them. If node \( i \) is node \( j \)'s parent, then the weight of node \( j \), \( w_j \), is given by \( w_j = (U_j(t) - U_i(t)) \cdot r_{ji} \), where \( r_{ji} \) is the transmission rate from node \( j \) to node \( i \). The transmission rate \( r_{ji} \) is the inverse of the time taken to transmit a packet successfully from \( j \) to \( i \). Hence it is dependent on the transmission power of node \( j \), and the interference between node \( j \).
and node \( i \). The transmission rate \( r_{ji} \) is maintained as an exponential weighted moving average which is updated every time a packet is successfully transmitted from \( j \) to \( i \).

The MDQ implementation can be performed in multiple ways. First, the maximum weight can be calculated in a 1-hop neighborhood or a 2-hop neighborhood. Second, if a node does not have the maximum weight in a neighborhood, it can modify both the initial back-off window \( (W_i) \) and the congestion back-off window \( (W_c) \), or just the congestion back-off window \( (W_c) \).

To cater for various combinations of the above design choices, we implement multiple versions of the MDQ MAC, each MAC is titled MDQ\( n \)-INIT\( m \) or MDQ\( n \)-CW\( m \). The variable \( n \) represents whether the calculation is performed in a 1-hop neighborhood or a 2-hop neighborhood (hence \( n = \{1, 2\} \)). For MDQ\( n \)-INIT\( m \), the max weight is calculated in a neighborhood of size \( n \) and if a node is not the maximum weight its initial and congestion back-off windows are increased by \( m \times W_i \) and \( m \times W_c \) respectively. For MDQ\( n \)-CW\( m \) if a node is not the maximum weight only its congestion back-off window is increased by \( m \times W_c \). We choose \( m \) to be either 2 or 4. It should be noted that the MDQ2-INIT\( m \) and MDQ2-CW\( m \) MAC are similar to the one proposed by Akyol et al. [1].

### 7.3.3 Implementing DQWM MAC

In order to implement the differential queue window mapping MAC, we need a function that can map the differential queue backlog to the initial back-off window \( W_i \). In [96], Warrier et al. present a heuristic to map the queue differential to a priority. They then use mechanisms provided by an 802.11e MAC to enforce this priority. Since the heuristic is specific to the 802.11e MAC, it is not portable to a generic CSMA MAC. In order to have a more systematic approach, to ascertain this mapping function, we use the work presented by Jiang et al. [37]. Jiang et al. [37] show that,
in a idealistic setting, under the assumption that a CSMA MAC is able to avoid all collisions, the back-off value used by a CSMA MAC should be an exponential random variable with rate \( \frac{Q(t)}{k} \). where \( k \) is a constant and \( Q(t) \) is the queue differential of the node at time \( t \). Therefore the average window size should be \( \frac{k}{e^{Q(t)/k}} \). Since the existing CSMA MAC performs a uniform back-off on a given window size, instead of choosing the back-off value as an exponential random variable with mean \( \frac{k}{e^{Q(t)/k}} \), we choose the window uniformly between \([0, \frac{k}{e^{Q(t)/k}}]\). The key property we have used from the work by Jiang et al. [37], is that the back-off window size decreases exponentially with the queue differential.

Note that the choice of \( k \) defines the back-off window size when the queue differential is zero, and hence determines how much a node with a small queue differential should be penalized, as compared to a node with a larger queue differential. Further as \( Q(t) \) increases, the initial back-off window exponentially decreases. Since in practice we have to have a minimum window size, we set the minimum value to 1 ms. The exponential decrease in the window size with the queue differential is also a reason why we chose to implement DQWM with the initial back-off window as compared to the congestion back-off window. The congestion back-off for the CC2420 CSMA is already set to 2.5 ms. Thus, if the max window size is set to 2.5 ms, by the time the queue differential becomes 2 we will be at the minimum window size.

Since the performance of the DQWM MAC depends on the parameter \( k \), we implement different versions of DQWM MAC, and refer to each version of the MAC by DQWM-\( k \). The different versions of the DQWM MAC are obtained by varying \( k \) from 10 ms to 10000 ms in multiples of 10.
7.3.4 Implementing PDQ MAC

The positive differential queue MAC is trivial to implement. In the PDQ MAC a node is allowed to inject a packet from its forwarding engine to the MAC, if and only if its queue differential is positive. The PDQ MAC thus does not perform any prioritization of node transmissions, and hence does not require to modify the MAC window sizes.

7.4 Is MAC layer queue prioritization necessary?

The first question we want to answer is whether the proposed queue prioritization techniques (MDQ, DQWM) improve the rate region of an existing CSMA MAC. If not, then the backpressure-based rate control protocol (BRCP) can be implemented using the simple PDQ MAC. In order to resolve this question, we evaluate the performance of various queue differential prioritization schemes (MDQ, DQWM) against the PDQ MAC. The different versions of the MDQ MAC (labeled either MDQ\textsubscript{n}-INIT\textsubscript{m} or MDQ\textsubscript{n}-CW\textsubscript{m}), and DQWM MAC (labeled DQWM-k) have been described in section 7.3. We implement the backpressure based rate control protocol (BRCP) with a log utility flow controller to run on top of different MAC schemes to present a comparative evaluation. This results in different implementation of BRCP for each MAC. We refer to each version by the name of the underlying MAC implementation. The metrics used to compare the different stacks are the total log utility and average total queue size. If the queue prioritization techniques (MDQ, DQWM) do improve the rate region of the existing substantially, we should see a much better log rate-utility performance for the MDQ/DQWM stacks, as compared to the PDQ stack.
We run each of these stacks on the 2 different topologies shown in Figures 7.3 and 7.4. The average link quality (PRR) in these topologies range from 40%-80%. All experiments are done using the USC Tutornet testbed [46]. To have a richer set of experiments we performed experiments on the 20 and 40-node topologies at two different power levels (power level 5 and 10). Figure 7.5 indicates the connectivity levels for both 20 and 40-node topologies under transmit power levels \{5, 10\}. 

Figure 7.3: The 20-node routing tree
7.4.1 Log utility flow controller

Design of the log utility flow controller follows the description presented in section 7.2.1. The log utility flow controller tries to maximize the global utility function $\sum g(r_i)$, where $g(r_i) = \log(r_i)$. Therefore, by equation (7.2), the instantaneous rate $R_i(t)$ is:

$$R_i(t) = \frac{V}{2U_i(t)}$$  \hspace{1cm} (7.3)

We show in the next section, how the parameter $V$ is chosen. Our choice of the log utility flow controller for performing this evaluation is motivated by the fact that in wired networks, maximizing the total log utility amounts to achieving proportional fairness [43]. Further, in a wireless setting, [66] shows that log utility presents the best trade-off between fairness and efficiency.
Figure 7.5: Connectivity for 20-node topology (Figure 7.3) and 40-node topology (Figure 7.4). Links indicate PRR of at least 80%.
7.4.2 Parameter selection

As described in section 7.2, the parameter $V$ presents a trade-off between the utility achieved and the average system queue size. In order to tune $V$ optimally for the PDQ MAC stack, we plotted the total log utility and average total queue size for the three different topologies shown in figure 7.6. For each topology the system utility increases with $V$ and saturates beyond a certain value. The queue size also increases with $V$. For the 20-node and 40-node topologies it is important to note that the system utility drops beyond a particular $V$. This is due to the fact that the maximum buffer size in these systems is finite (maximum of 70 packets, 10 byte payload each), and hence beyond a certain $V$ the required queue sizes cannot be supported. This results in packet drops, lowering total utility. Using Figure 7.6 $V = 150$, for the 20-node topology, and $V = 60$ for the 40-node topology.

Although the plots in figure 7.6 were obtained using the PDQ stack, we use the same value for all the other stacks as well. We believe this is a valid choice, as the log utility and average queue size of backpressure protocols will increase monotonically with $V$ [29] in the absence of buffer overflows. When comparing stacks with identical $V$, the stack that outperforms will have
an equal or better utility for an equal or smaller queue size. If a stack has both lower utility and smaller queue size, we will need to increase $V$, in order to have a fair comparison.

### 7.4.3 Comparing PDQ with MDQ and DQWM MAC

Figures 7.7 and 7.8 presents the total log utility and the average total queue size for the different stacks. The packet size in the experiments was 40 bytes, and the experiments were performed at two different power levels on each of the two topologies. The CC2420 radio allows 31 different power levels, 0 being the lowest and 31 being the highest. We choose a power level of 5 to represent a low interference scenario and a power level of 10 to represent a high interference scenario.

![Figure 7.7: Log utility performance of different MAC protocols across different topology sizes, and different power levels.](image)

Figure 7.7: Log utility performance of different MAC protocols across different topology sizes, and different power levels.

The log utility performance in Figure 7.7 shows interesting behavior across topologies. The total log utility decreases from the 20-node to the 40-node topology. Additionally, for 40-nodes, log utility decreases when the power level is increased. The reason for higher log utility for 20-nodes is that the rates for all sources remain greater than 1. For 40-nodes, due to reduction of available per flow capacity, a subset of sources get rate less than 1, leading to negative utility.
The sum log utility for 40-nodes is thus less than that for 20-nodes. For 40-nodes, the reduction of log utility due to increase in power level results from the increase of interference, visible in Figure 7.5. This results in reduced available capacity and hence leads to smaller total log utility.

Figure 7.8: Average total queue length for different MAC protocols across different topology sizes, and different power.

In the 20-node topology, the PDQ stack performs similar or better than all version of the MDQ stacks, in terms of total log utility as well average total queue size. This is true for both power levels. For the 20-node topology at power level 5, the DQWM-100 MAC presents the best performance in terms of log utility. However the log utility performance is only \(3\%\) higher than the PDQ MAC. All other versions of DQWM MAC perform worse than the PDQ MAC. Even for the DQWM-100 MAC, although the log utility is slightly better than the PDQ MAC, its average queue length is much larger than the PDQ MAC (by \(\sim100\) packets), implying that a similar performance can be achieved by PDQ MAC by increasing \(V\). For the 20-node topology at power level 10 the PDQ MAC performs better than all other version of MDQ and DQWM MAC.

For the 40-node topology, again the MDQ1-CW2 MAC and the MDQ2-CW2 MAC outperform the PDQ MAC only by a small margin. In this topology all versions of the DQWM MAC perform much worse than the PDQ and MDQ MAC. In terms of log utility the MDQ1-CW2
MAC and MDQ2-CW2 outperform the PDQ MAC by \( \sim 0.5 \) and total queue sizes are reduced by \( \sim 10 \) packets. Note that \( \sum \log(r_i) = \log(\prod r_i) \). Therefore if the performance gap of the log utilities is \( 0.5 \) this implies \( \frac{\prod_{MDQ1-CW2} r_i}{\prod_{PDQ} r_i} = e^{0.5} \sim = 1.5 \). If all of this rate gain were allocated to one of the 39 sources, that source will have rate 1.5 times greater. Given that the rates in these networks are on the order of \( \sim 1 - 2 \) packets/sec (for a 40-node topology), a factor of 1.5 for a single source will not significantly alter the rate distribution in the network. In terms of queue sizes, the difference in average per node queue size amounts to an increase of \( \frac{10}{39} = 0.25 \) packets under PDQ MAC, as compared to the MDQ1-CW2 MAC.

We believe that the small packet sizes (40 bytes) existing in these networks are a key reason for the comparable performance of PDQ, MDQ and DQWM Macs. The transmission times for these packets are usually smaller than the back-off window size (e.g., for TinyOS-2.x CSMA the packet transmission time for 40 byte packet over 250 kbps radio is 1.5 ms, while the congestion window is 2.5 ms). Since multiple transmissions can take place within a single window size, increasing the window size (as these heuristics do) might result in unnecessary loss of throughput. Further the impact of increasing the initial back-off (10 ms), as compared to the congestion back-off (2.5 ms) is much more adverse. This can be clearly seen in the superior performance of the MDQ_{n-CW^m} (which modifies only the congestion window) as compared to the MDQ_{n-INIT^m} (which modifies the initial and congestion window), as well as the DQWM MAC (which modifies only the initial back-off). Further, the MDQ1-CW^m outperforms the MDQ2-CW^m, implying that performing maximum weight calculations in a 2-hop neighborhood is unnecessary.

The key implication of the results presented in figures 7.7 and 7.8 is that the PDQ scheme performs comparable to the various MDQ and DQWM MAC scheme across different size topologies, and for different power levels. From the description presented in section 7.3, it is clear that
the complexity introduced by the MDQ and DQWM mechanism is not comparable to the gains presented by these modifications. In a sensor network setting where packet transmission times are much smaller than the back-off windows, backpressure-based protocols can be implemented with similar performance over existing CSMA MAC by simply implementing a PDQ MAC.

7.5 Sensitivity of transport layer to parameter settings

Figure 7.9: Static flow scenario goodput for IFRC and backpressure based stack 20 and 40-node topologies (Figures 7.3 and 7.4).
Our second goal is to evaluate the performance sensitivity of the backpressure-based rate control protocol (BRCP) under a fixed $V$ parameter in a dynamic flow scenario. A dynamic flow scenario is one in which the number of active flows varies over time. For the backpressure rate control stack, we choose two versions: one running over the PDQ MAC, and one running over the MDQ1-INIT4 stack. We choose these two MAC implementations because they are two extremes of the variants that we evaluated in section 7.4. As was seen in section 7.2.1, the only parameter that the flow controller performance depends on is $V$. In order to gage the resilience of this fixed parameter, we compare the different versions of BRCP (referred to as PDQ stack and MDQ1-INIT4 stack) against the state of the art rate control protocol in wireless sensor networks, namely the Interference Aware Rate Control Protocol (IFRC [67]).

IFRC [67] is an additive increase multiplicative decrease (AIMD) protocol that attempts to achieve lexicographic max-min fairness [4] over a collection tree. IFRC is a distributed, router centric, rate control mechanism that detects congestion using a preset queue threshold and sends explicit congestion notification to the set of nodes it deems responsible for causing congestion, asking them to perform multiplicative decrease on their rates.

We use the 20 and 40-node topologies in order to perform a comparative evaluation between the backpressure-based rate control stacks and IFRC. We consider two scenarios of traffic flow on these topologies. All nodes except the root (node 12 in 20-node topology, and node 29 in 40-node topology) are considered to be sources. We define a static scenario as one in which all flows are active for the entire duration of the experiment.

As IFRC aims to achieve lexicographic max-min fairness, a valid comparison cannot be achieved using the log utility flow controller described in section 7.4.1. Instead we design a

---

IFRC is also described in Chapter 5, Section 5.6.1
Figure 7.10: Static flow scenario queue length comparison for IFRC and backpressure-based stack on 20 and 40-node (Figures 7.3 and 7.4) topologies.

new flow controller using the notion of $\alpha$-fairness. We describe the design of the $\alpha$-fair controller before presenting our comparative results.

### 7.5.1 $\alpha$-fair controller

The utility function for $\alpha$-fairness is given by $g(r_i) = \frac{r_i^{1-\alpha}}{1-\alpha}$. The total utility is therefore:

$$\sum_{\forall i} \frac{r_i^{1-\alpha}}{1-\alpha}$$

(7.4)
Here, $\alpha$ is a constant greater than 1. Theoretically, it has been shown that when $\alpha \to \infty$, $\alpha$-fairness approaches lexicographic max-min fairness [66].

Given that the $\alpha$-fair objective is defined by equation (7.4), substitution into equation (7.2) results in a flow controller that will set its instantaneous rates based on the following equation:

$$R_i(t) = \alpha \sqrt{\frac{V}{U_i(t)}}$$

(7.5)

When $U_i(t) = 0$ we set $R_i(t) = \sqrt{V}$. Ideally when queues are zero, $R_i(t)$ should be set to $R_{max}$, but since $R_{max}$ is unknown we set it to $\sqrt{V}$. In order to achieve lexicographic max-min fairness we want $\alpha$ to be large. We are able to achieve results comparable to IFRC for our specific sensor network setting with $\alpha = 5$.

### 7.5.2 Comparing backpressure and IFRC

The queue threshold we use for IFRC is 20 packets. The parameters for the backpressure stack were chosen by doing multiple static flow runs over the 20 and 40-node topologies while varying $V$. The fixed parameter value that provided goodput comparable to IFRC was a setting of $V = 610$ for the 20-node topology and $V = 3$ for 40-nodes. This resulted in an average per-node queue size of approximately 9-10 packets under the backpressure stacks.

### 7.5.3 Static case

Figures 7.9(a) and 7.9(b) show the static flow goodput performance of the PDQ stack, MDQ1-INIT4 stack, and IFRC over the 20 and 40 node topologies. We present the results for the static scenario to justify our flow controller implementation. As can be seen in Figures 7.9(a) 

---

*We avoid using an arbitrarily high value to avoid queue drops.*
Figure 7.11: Behavior of PDQ and IFRC under dynamic flow scenario for 20 and 40-node topologies with a fixed $V$.

and 7.9(b), the rate vector presented by the backpressure stacks is lexicographically greater [4] than the rate vector presented by IFRC. Thus, for the static scenario the backpressure stack is able to present better max-min fairness than IFRC. The throughput improvements for the 20-node network ranges from 1.5x-2x, and for the 40 node it ranges from 1.5x-2.5x.

We believe there are three possible reasons for the superior throughput performance of the backpressure stack: first, with an optimal setting of $V$, the backpressure stack can ensure that the system operates very close to the achievable rate region of the network. IFRC on the other hand uses an AIMD scheme operates at an average source rate at $\frac{3}{4}$th the achievable capacity [67]. Second, using the PDQ MAC may result in an improvement in the network rate region, resulting
in a max rate for backpressure greater than what IFRC is observing. Finally, backpressure does not assume any graph theoretic model, while the signaling mechanism of IFRC assumes a graph theoretic notion of interference which is conservative potentially resulting in lower throughput. Though the backpressure stack outperforms IFRC in terms of rates, it does at the cost much larger queue backlogs (and hence end-to-end delays). This can be observed in Figure 7.10.
7.5.4 Dynamic case

We now evaluate the performance under a dynamic flow scenario. To generate a dynamic scenario on the 20 and 40-node topologies we use the following flow activation strategies. In the 20-node topology, nodes \{1, 7, 13, 11\} are active for the complete duration of the experiment while all other nodes are activated at 500 seconds into the experiment. For the 40-node topology, nodes \{20, 21, 30, 31\} are active for the entire duration of the experiment while all other flows are activated at 500 seconds into the experiment. Each of these experiments last for a duration of 1000 seconds.

Figure 7.11 shows the behavior of the PDQ stack and IFRC, in a dynamic setting, with a fixed V parameter. Note that the y-axis in each of these graphs is in log scale. For both topologies, it can be seen that when a few flows are operational in the network, the goodput given to these flows is much higher in the case of IFRC as compared to the PDQ stack. This can be seen between 0 – 500 seconds for both the 20 and 40 node topology. When all flows become active the scenario becomes the same as the static case, and as seen before PDQ outperforms IFRC. Due to space constraints, MDQ1-INIT4 graphs are not presented. But as seen from the goodput in Figure 7.14, the performance of MDQ1-INIT4 is similar to PDQ.

The above behavior is an artifact of the fixed V setting. For the 20-node topology $V = 610$. A node’s transmission rate is maximized when the local queue is empty, as per Equation 7.5. The maximum rate a node can achieve in the 20-node topology is therefore $\sqrt[3]{610} = 3.60$ pkts/sec. However, as can be seen from Figure 7.11(b), when only 4 flows are active they can potentially achieve 20 packets/sec. Thus the fixed setting of V forces the flows to under perform. We cannot enlarge V here because this will result in queues overflowing once all traffic in the network is
Figure 7.13: Behavior of PDQ under dynamic flow scenario for 20 and 40 node topologies with an adaptive $V$.

active (recall that this $V$ resulted in average per node queue sizes of 20 packets under our static flow tests). A constant setting of $V$ therefore has to cater to the worst case scenario. The same arguments apply for the 40-node scenario.

Given the failings of BRCP to outperform IFRC in a dynamic flow scenario, we argue that adapting the parameter $V$, with the number of active flows in the network, is essential to exhibit the desired gains. To validate this argument we ran the same dynamic flow experiment on the 20 and 40-node topologies with an adaptive $V$. The protocol sets $V = 300000$ when there are only
Figure 7.14: Goodput performance of IFRC, PDQ and MDQ MAC on 20 and 40 node topologies, under dynamic flow scenario, with *adaptive* $V$.

4 flows existing in the network, and sets $V = 610$ (20-node topology), and $V = 3$ (40-node) topology respectively when all flows are active. The rate allocation behavior with the *adaptive* $V$ setting, for the PDQ stack, are shown in Figure 7.13. Clearly with a larger $V$, when only 4 flows are active a higher rate can be achieved by the 4 flows as compared the *fixed* $V$ setting. The effects of the *adaptive* $V$ can be seen in the goodput plots as well (Figure 7.12). Thus, with an *adaptive* $V$ BRCP clearly outperforms IFRC. The throughput improvements with the *adaptive* $V$ ranges from 1.5x-2.5x for the 20-node topology, and 1.5x-3.5x for the 40-node topology.
7.6 Summary

In this chapter we undertook the first exhaustive empirical evaluation of the backpressure-based rate control (BRCP). We have shown that in a sensor network setting, a pure backpressure approach (PDQ) performs comparably to schemes that attempt differential queue prioritization (MDQ and DQWM) as a means for approximating optimal backpressure scheduling. This implies that the backpressure-based rate control protocol (BRCP) can be developed on existing CSMA MAC without modifications. We also show that automatic parameterization is necessary for BRCP to perform well in dynamic flow scenarios.

Although the empirical results were presented for a collection tree, we believe they hold for a general any-to-any backpressure implementation. As shown by Akyol et al. [1], support for any-to-any traffic is possible through the addition of per destination queues. For the single destination case, we reason that PDQ is performing as well as MDQ MAC due to the small packet sizes that exist in wireless sensor networks. Thus, even with addition of per destination queues, the comparative results we are observing should hold. Further, by increasing the number of supported destinations, the queue behavior with $V$ will remain the same, and hence our results on the requirement of automatic parameterization in a dynamic flow setting still hold.
Chapter 8

Conclusions and Future Work

In Chapter 5, we presented the design and implementation of the Wireless Rate Control Protocol (WRCP), which is the first protocol to use explicit feedback based on precise capacity information to achieve a max-min fair rate allocation over a collection tree. Through a comparative evaluation with IFRC [67], the state of the art in congestion control in WSN, we have shown the advantages of using explicit capacity information in designing rate control algorithms when compared to AIMD approaches.

WRCP shows considerable gains in terms flow completion times as compared to IFRC, and approximately a factor of $1.5 - 2$ reduction in end-to-end packet delays. The key gains of using WRCP over IFRC occur when there are a large number of short flows in the networks. In this scenario, WRCP correctly allocates a better max-min fair solution to the flows in the networks, as compared to IFRC, leading to faster flow completion times. The faster flow completion times indirectly lead to higher energy savings. Chapter 5 also highlighted the versatility of WRCP, in the form of mechanisms built into WRCP, that can make the protocol work in the presence of external interference as well. As with any engineering solution, WRCP has its limitations as well. As compared to rate control protocols such as IFRC, which rely on AIMD-based mechanisms
to estimate available network capacity, WRCP relies heavily on the receiver capacity model to estimate available capacity. This in turn limits the CSMA MACs on which WRCP can operate, since the values of the receiver capacity are currently relevant to the CC2420 CSMA MAC. If WRCP needs to be made operational on other sensor network MACs, then the value of receiver capacity will need to be recalculated for those MAC protocols.

In Chapter 6, we presented the first Backpressure-based Rate Control Protocol (BRCP) for wireless sensor networks. Using the receiver capacity model we were able to adapt the stochastic optimization technique presented by Neely et al. [58], to present a back pressure based rate control stack for TinyOS-2.x on the Tmote sky platforms. In Chapter 7, we also presented versions of the implementations that did not require information based of the receiver capacity model, namely PDQ (positive differential queue), MDQ (maximum differential queue prioritization), and DQWM (Differential Queue Window Mapping). The MDQ and DQWM MAC’s are heuristics that try to emulate the ideal backpressure scheduler proposed by Tassiulas et al. [89], while the PB scheme is a naive back pressure implementation that simply performed control actions based of differential queue backlogs, agnostic of the neighboring queue differentials. In Chapter 7 we explored the design space of the backpressure based rate control protocol, to show that we do not need to implement the complicated heuristics such as MDQ and DQWM to get good performance out of the backpressure-based protocol. A naive algorithm, such as the one implemented in PDQ MAC, suffices to present considerable gains over existing rate control protocols. Further, we also showed that there exists a gap between theory and practice when it comes to implementing backpressure-based protocols over real systems. The gap manifests itself in the form of the parameter setting $V$ at the transport layer, that controls the trade-off between optimal rate
allocation and delay. Our empirical analysis in Chapter 7 shows that in order for backpressure-based stacks to outperform existing protocols, the parameter $V$ has to be adapted to the network topology and the number of flows in the network, in order to provide the desired performance gains over dynamically changing networks.

8.1 Future work

In this work, our main focus has been on designing transport layer rate control mechanisms for wireless sensor networks. Our investigation into the design of flow controllers for sensor networks using backpressure-based techniques, has sparked several ideas that can form the basis for future research. First, to investigate design of backpressure-based routing protocols; second, investigating online algorithms that will allow for the automatic adaptation of the parameter $V$, that is critical to the performance these backpressure-based protocols, based on the network topology and number of flows existing in the network; third, to investigate mechanisms that can be implemented over the CSMA-CA MAC in order to improve rate-utility performance of backpressure-based stack; and fourth, to extend the cross-layer stacks implemented in this thesis to incorporate information from the physical layer as well, in order to implement physical layer technologies such as cooperative communication and analog network coding over multi-hop wireless networks. We elaborate on each of these below.

8.1.1 Improving backpressure scheduling over CSMA

In Chapter 7, we showed empirically that heuristics that employ complex queue prioritization at the MAC layer (such as MDQ and DQWM MAC) are not able to outperform a naive backpressure
scheduling mechanism such as the PDQ MAC, that employs a very coarse grained form of queue prioritization (it allows packet forwarding if the queue differential is positive). A reason for the lack of gain in rate-utility performance, for techniques such as MDQ and DQWM could be the sub-optimal mapping of mapping of the queue differential to the back-off window size of the CSMA MAC. A potential research area is therefore to develop theory that can rigorously quantify the mapping between the queue differential and back-off window size in an existing CSMA-CA MAC. This research will aim to answer the following question: Does an optimal mapping exist between the queue differential and the back-off window size of a CSMA MAC, that can outperform the PDQ MAC? An affirmative answer to this question will lead to the question of finding the mapping, and developing a MAC that will exhibit the gains absent in the MDQ and DQWM implementations.

8.1.2 Online algorithms for dynamically adapting $V$

In Chapter 7, we showed empirically that a fixed value of the parameter $V$ will result in serious underperformance of the rate control protocol, when the network is experiencing dynamics in terms of the number of flows in the network. While this problem was explored specifically in the context of transport layer rate control design, we conjecture that it will exist in any backpressure-based protocol, that might be providing functionality at different layers of the stack and has been designed using the generic cross-layer optimization framework presented by either Neely et al [58] or Stolyar [83]. Therefore to make backpressure-based protocols viable in general, it is imperative to design online algorithms that can dynamically adapt $V$, based on the network configuration (topology, number of flows, etc.). One possible step in this direction could be design an AIMD mechanism that probes the network to ascertain the possible value of $V$. 

164
8.1.3 Backpressure-based routing protocols

The main application of backpressure-based techniques in this thesis has been in designing flow controllers. Using a backpressure based stack results in the queues presenting information about the cost of sending a packet over a particular route. For designing the flow controllers this cost was interpreted as the amount of congestion that a source causes in sending packets into the network. This cost could also be interpreted as path quality, and hence the same backpressure gradients can be used discover paths to the sink. The idea of using backpressure-based techniques for designing routing protocols is not new, and, at least from a theoretical perspective, has been explored before [29]. The backpressure-based routing protocols proposed are elegant in design, and can perform routing even in mobile networks [59]. Also, theoretically, they can result in achieving the best possible rate region. However, from a systems perspective, the key problem with backpressure-based routing is that they tend to form transient loops in the network; packets keep traversing these loops till enough backpressure gradient is built to route the packets out of the network, resulting in higher end-to-end packet delays, and larger expenditure on energy for delivering a packet from a source to the sink, when compared to shortest path routing (such as Collection Tree Protocol (CTP) [26]). The design and implementation of a backpressure-based routing protocol is still simple and elegant enough to warrant a more thorough examination and comparison with shortest path routing protocols such as CTP (there are no known backpressure-based routing protocols for sensor networks). This investigation will firstly quantify the drawback of a backpressure-based routing scheme; secondly it will help us gain insights for designing mechanisms that might reduce the inherent loops and delay, due to the use of backpressure-based techniques, possibly resulting in an improved backpressure-based routing protocol.
8.1.4 Enabling interference cancellation techniques over multi-hop

A key insight gained by the empirical evaluation of rate control protocols on real sensor networks testbeds is the drastic reduction of available capacity with increase in the number of nodes, and the number of hops in the network (Chapter 1). The reduction in capacity is primarily due to the use of physical and link layer technologies that were designed for single hop networks, which exhibit poor performance in a multi-hop setting. The drastic reduction of capacity with increase in network size suggests, that with current wireless link layer technology it might not be possible to make large scale (consisting of a few thousands of nodes) multi-hop wireless networks viable in practice.

Given the poor performance of traditional link/physical layer technologies with increasing number of interferers (senders), communication theorists have shown that by designing new link/physical layer that do not try to avoid interference, but in fact encourage ‘constructive’ interference, the capacity of a wireless network can be improved drastically in the presence of increasing number of interferers. Examples of such link/physical layer technologies are analog network coding [42] and cooperative communication [60]. Using software defined radios, researchers have also demonstrated the viability of developing practical systems using these technologies in simple single-hop settings.

A common objective for these new class of link/physical layer technologies is to find a set of simultaneous interfering senders. The selection criteria can be based on choosing interfering senders, such that the senders can cooperate to generate ‘constructive’ interference at the relay node (amplifying the signal strength) [60]; or, if analog network coding [42] is being used, the selection of interfering senders can be such that the receivers overhear information about the
interfering senders, allowing the relay to perform analog network coding [42] on the interfering signals resulting in simultaneous transmissions. In a multi-hop setting, in order to maximize the number of simultaneous transmissions, selection of senders will depend on the set of nodes that currently have data to transmit, which in turn depends on the routing decisions. Thus, in order to optimize the rate region presented by these cooperative link/physical layer, routing and link layer protocols need to be designed by solving a joint routing/scheduling problem. Beyond routing and link/physical layer design, there also remains the question of designing transport protocols for these networks. Given the diverse nature of applications targeted for these networks, no single rate-based utility function will satisfy all applications. Hence transport protocols for these networks should have the ability to solve for application defined utility functions. Achieving the later implies solving a joint scheduling, routing and transport problem. A research goal, in this area, is therefore to leverage the cross-layer design experience in this thesis to re-engineer the network stack; by designing practically viable cross-layer design of cooperative transport, routing and MAC protocols which will help realize the vision of large scale, multi-hop wireless networks.
References


