ABSTRACT

From a theoretical standpoint, backpressure-based techniques present elegant cross-layer rate control solutions that use only local queue information. It is only recently that attempts are being made to design real world wireless protocols using these techniques. To aid this effort, we undertake a comprehensive experimental evaluation of a backpressure-based network stack over the USC tutornet wireless testbed, a 100 node wireless sensor network (WSN) testbed spanning two floors. To the best of our knowledge this is the first such study in the context of a WSN. Our evaluation yields three key insights. First, we show that in a WSN, contrary to previous proposals, the gains in implementing queue prioritization over a CSMA MAC are negligible. Hence backpressure-based protocols can be implemented for a WSN without modifying the underlying CSMA MAC. Second, we show that the performance of backpressure-based protocols is highly sensitive to a parameter setting that depends upon current traffic conditions. Therefore, practical backpressure protocols must provide for automatic parameter adaptation. We conjecture that this insight, though presented here in the context of a WSN, holds for any general multi-hop wireless network. Third, our comparative evaluation with existing rate control protocols shows that with optimal parameter settings, backpressure-based stack can give up to a factor of two improvement in throughput performance, albeit at the cost of increased queue sizes.

1. INTRODUCTION

In wireless sensor networks, in order to avoid congestion collapse, it is imperative to design rate control mechanisms as part of the communication stack. Even though applications targeted towards sensor networks have low data rates, the drastic degradation in available source rate\(^1\), with increase in the size of the network, can force these low data rate applications to cause congestion collapse when they are deployed in networks of larger scales. Presence of rate control mechanisms will keep the source rates well informed of the achievable rate, and avoid congestion collapse\(^2\). A key challenge in designing rate control stacks for wireless sensor networks is the application diversity in these networks. For this reason, rate control stacks need to be flexible enough to optimize for different types of rate-based utility, which can cater to application specific requirements.

In the past few years, a novel theoretical framework developed by Neely et al.\cite{16} and Stolyar\cite{23} give elegant backpressure-based stochastic rate control mechanisms that maximize convex rate-utilities while ensuring queue stability. The solutions from this framework result in modular decisions that can be implemented at the transport, network and MAC layer; presenting an elegant design framework for realizing flexible rate control stacks in a wireless setting. Given the elegance of these results there have been attempts in the 802.11 multi-hop network setting to translate these theoretical solutions into implementable protocol stacks, which can operate over CSMA based wireless networks (\cite{3,18,26}).

Given the existing proposals for backpressure-based stacks over 802.11 networks, we consider the implementation of such a stack for data-gathering in a 802.15.4-based wireless sensor network. Though backpressure based techniques can allow for routing decisions as well, we focus our efforts on discerning MAC and transport layer issues assuming shortest path routing. We perform this evaluation by implementing a backpressure-based rate control stack, as part of the TinyOS-2.x communication framework, and evaluate the stack over the USC tutornet testbed\cite{12}. This is a 94 node wire-
less sensor network (WSN) spanning two floors. The testbed is composed of Tmote sky devices [1]. To the best of our knowledge, this is one of the first evaluations performed in a WSN context. Our evaluation helps us garner insights into design issues related to MAC layer modifications, and transport layer parameter settings which are required to simplify backpressure-based rate control implementation, and improve its rate-utility performance 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 proposals ([3, 26]) try to emulate the optimal backpressure scheduling policy [24], 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 give links 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. It is unclear if the complicated queue prioritization techniques proposed actually improve the existing rate region in practice. 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 [23] or Neely et al [16], 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 trade-off between queue size and rate-utility performance ([3, 16]). For these protocols, rate-utility performance improves logarithmically with increase in queue size [16]. 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 with existing rate control protocols, irrespective of the number of active flows in the system?

Our contributions in this work are three-fold:

- First, a comparative evaluation of different heuristics for backpressure scheduling over a CSMA MAC shows that in a sensor network setting, backpressure-based protocols 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 a state of the art rate control protocol (IFRC [20]) 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 protocols 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 protocols can easily outperform existing rate control stacks, by as much as a factor of 2.

The paper is organized as follows; in section 2, we present related work. In section 3, we present a software architecture that captures the general design of a backpressure based rate control stack. In section 4, we present the implementation details of heuristics that have been proposed for implementing backpressure scheduling over a CSMA stack. In section 5, 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 6, we present an evaluation of the backpressure-based rate control protocol against IFRC in order to understand the parameter dependence of backpressure protocols. In section 7, we present a summary of our results and future directions for this work.

2. RELATED WORK

From a theoretical standpoint, the formulation of the cross-layer rate control stack design as a convex optimization problem was motivated by the seminal works

\( ^3 \)The notation \( V \) for this parameter has been borrowed from the work by Neely et al [16], the work by Stolyar [23] introduces a similar parameter called \( \beta \).
The work by Radunovic et al. [18], and Low and Lapsley [13]. These works also promoted the use of duality-based techniques for formulating distributed solutions to this convex optimization formulation. Neely et al. ([14, 15, 16]) and Stolyar [23] have further strengthened these theoretical foundations, by presenting mathematical frameworks that can solve the convex constrained formulation in a stochastic setting, in a distributed manner. The ability to solve these problems in a stochastic settings makes the solutions proposed by Neely et al. and Stolyar particularly relevant in a wireless multi-hop setting. Both these frameworks rely heavily on the existence of a back-pressure scheduling policy to be implemented at the MAC layer. The back-pressure scheduling policy was proposed by Tassiulas et al. [24], and is known to be throughput optimal. While much of the theoretical work on stochastic network optimization assumes TDMA scheduling using maximum weight matching, a recent result by Jiang and Walrand [9] shows that under idealized conditions optimal solutions can also be obtained by a CSMA MAC which locally prioritizes links with higher queue differentials.

Several recent proposals have attempted to implement backpressure scheduling on practical 802.11 based wireless networks. Warrier et al. [27] and Akyol et al. [3] propose schemes which try to achieve probabilistic prioritization of the node transmissions by modulating the MAC contention window size based on a nodes’ queue differential with its parent. Akyol et al. [3], also proposes design of flow controllers on top of these schedulers based on the technique proposed by Stolyar [23]. The work by Radunovic et al. [18] develops a multi-path routing and rate control protocol, that can be integrated with TCP over 802.11, using backpressure techniques. They use a simple backpressure scheduler that allows transmissions as long as the queue differential is greater than a threshold. None of these proposals are able to divine which backpressure scheduler heuristic should be used to give the best performance in a given setting. Further, since these works target 802.11 networks, their comparison is with TCP which is known to perform poorly over wireless [4]. We believe this lack of evaluation with protocols that have been optimized over wireless hides the parametric dependence of back-pressure protocol performance.

There have been several proposals addressing the problem of rate control in wireless sensor networks ([6, 8, 17, 20, 21, 25, 28]). Most of these protocols have assumed a clean slate design and follow a router centric, explicit congestion notification approach. A key failing of these proposals, when compared to the backpressure based stacks, is that these protocols are monolithic in nature, optimizing for a specific rate-utility function, or performing pure congestion control without regard to utility optimization. Given the diversity of applications targeted towards sensor networks, it is unclear as to which utility optimization (proportional fairness, max-min fairness, throughput maximization) will best serve the requirements of an application. Given these diverse applications, the ability of backpressure protocol stacks to optimize for any type of concave rate-utility function presents a clear advantage against existing rate control mechanisms. Despite the monolithic nature of existing protocols, resulting in lack of flexibility required by sensor network applications, these protocols can act as good benchmark to calibrate backpressure stack performance, since they have been optimized to perform in a multi-hop wireless setting. Amongst these protocols we choose IFRC [20], which strives to achieve lexicographic max-min fairness, as a benchmark to backpressure protocol performance.

3. BACKPRESSURE-BASED STACK

We start our investigation by presenting a software architecture (Figure 1) that captures the design of a generic backpressure based rate control stack. We re-

![Figure 1: Software architecture for a backpressure-based stack.](image-url)
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 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 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.

We now describe the implementation of the transport and MAC layer in further detail.

### 3.1 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 \( g(r_i) \), such that the rate allocation \( r_i \), \( \forall i \), maximizes \( \sum g(r_i) \) across the entire network. \( g(r_i) \) 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 (\( \max \left( \sum_{i} g(r_i) \right) \)). In order to design such flow controllers we can use one of two techniques presented by Stolyar [23] and Neely et al. [16].

In the proposal presented by Sridharan et al. [22], the flow controller is designed using a technique proposed by Neely et al. [16]. 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:

\[
\max \left[ \frac{V}{2} \cdot g\left(R_i(t)\right) - U_i(t) \cdot R_i(t) \right]
\] (1)

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

\[
\frac{V}{2} \cdot g'(R_i(t)) = U_i(t)
\] (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 \( g(r_i) \). A large value of \( V \) will imply large value of \( U_i \), and large \( g(r_i) \). Whereas a small value of \( V \) will imply small value of \( U_i \), and small \( g(r_i) \).4

### 3.2 Backpressure-based MAC

![Diagram showing backpressure scheduling](Image)

Figure 2: Understanding backpressure scheduling.

Ideally, a backpressure-based MAC should implement the scheduling policy proposed by Tassiulas et al. [24]. Figure 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 contention 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 weight matching schedule, in practice this 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 ([3, 26]) 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. [3] achieve 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. [3] as the Max Differential Queue MAC (MDQ).

Warrier et al. [26] achieve queue prioritization by having queue differential thresholds mapped to correspond-

---

4It should be noted that flow controllers in the proposal by Akyol et al. [3], are very similar in structure to the one shown in equation 2. The only difference between the two designs is the parameter \( \beta \), which has an inverse effect as compared to \( V \).
ing window sizes. When a node’s 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).

In addition, we also consider a simpler approach similar to the technique employed by Radunovic et al. [18]. 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.

4. IMPLEMENTING BACKPRESSURE

The target platform for presenting our evaluation of backpressure based protocols in WSN is the Tmote sky device [1]. 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 [3, 26] (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.

4.1 The CC2420 CSMA MAC

The CSMA-CA algorithm in CC2420 CSMA MAC [2] operates on only 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 × 32.25) initial back-off window, and 2.58 ms (80 × 32.25) congestion back-off window.

4.2 Implementing MDQ MAC

The maximum differential queue prioritization technique (MDQ), proposed by Akyol et al. [3], and described in section 3.2, 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\( m \)-INIT\( m \) or MDQ\( m \)-CWM. 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\( n \)-CWM 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. [3].

4.3 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 [26], 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. However, Warrier et al. do not present any justification in [26] for the use of the proposed heuristic, and hence the heuristic is not directly portable to any other CSMA MAC. In order to have a more systematic approach, we use the work presented by Jiang et al. [9]. This work shows that, under the idealistic 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 \( \frac{1}{k} \). 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 \( \frac{1}{k} \), we choose the window uniformly between \([0, \frac{1}{k}]\). The key property we have used from the work by Jiang et al. [9] 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.

### 4.4 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 need modification of MAC window sizes.

### 5. 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 stack 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. We implement a back-pressure based rate control stack with a log utility flow controller to run on top of different MAC schemes to present a comparative evaluation. The different versions of the MDQ MAC (labeled either MDQ\( n \)-INIT\( m \) or MDQ\( n \)-CW\( m \)), and DQWM MAC (labeled DQWM-\( k \)) have been described in section 4. 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.

![Figure 3: The 20-node routing tree](image3.png)

We run each of these stacks on the 2 different topologies shown in Figures 3 and 4. The average link quality (PRR) in these topologies range from 40%-80%. All experiments are done using the USC Tutorials testbed [12]. 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 5 indicates the connectivity levels for both 20 and 40-node topologies under transmit power levels \{5, 10\}.

### 5.1 Log utility flow controller

Design of the log utility flow controller follows the description presented in section 3.1. 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

![Figure 4: The 40-node routing tree](image4.png)
This is due to the fact that the maximum buffer size that the system utility drops beyond a particular value. The queue size also increases with utility increases with

\[
V = 150, \quad \text{for the 20-node topology, and } V = 60 \quad \text{for the 40-node topology.}
\]

Although the plots in figure 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 \( V \) [7] 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.

### 5.3 Comparing PDQ with MDQ and DQWM MAC

Figures 7 and 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.

The log utility performance in Figure 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 reach 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 5. This results in reduced available
than the PDQ MAC (by the PDQ MAC, its average queue length is much larger than the 100 MAC, although the log utility is slightly better than the PDQ MAC. Even for the DQWM-100 MAC, although the log utility performance is only 3% higher than the PDQ MAC. All other versions of DQWM MAC perform much worse than the PDQ MAC. All other versions of MDQ and DQWM MACs are comparable to the various MDQ and DQWM MAC scheme across different size topologies, and for different power levels. From the description presented in section 4, 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 then the back-off windows, backpressure based protocols can be implemented with similar performance over existing CSMA MAC by simply implementing a PDQ MAC.

6. HOW SENSITIVE IS TRANSPORT LAYER TO PARAMETER SETTINGS?

Our second goal is to evaluate the performance sensitivity of a backpressure-based rate controller under a fixed V parameter in a dynamic flow scenario. A dynamic flow scenario is one in which the number of ac-

Figure 7: Log utility performance of different MAC protocols across different topology sizes and different power levels.

Figure 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 ~100 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 ~ 0.5 and total queue sizes are reduced by ~ 10 pack-
tive 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 5. As was seen in section 3.1, the only parameter that the flow controller performance depends on is $V$. In order to gauge the resilience of this fixed parameter, we compare the backpressure rate control stack against the state of the art rate control protocol in wireless sensor networks, namely the Interference Aware Rate Control Protocol (IFRC [20]).

IFRC [20] is an additive increase multiplicative decrease protocol that attempts to achieve lexicographic max-min fairness [5] 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 rate control stack 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 5.1. Instead we design a new flow controller using the notion of $\alpha$-fairness. We describe the design of the $\alpha$-fair controller before presenting our comparative results.

### 6.1 $\alpha$-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:

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

(4)

Here, $\alpha$ is a constant greater than 1. Theoretically, it has been shown that when $\alpha \rightarrow \infty$, $\alpha$-fairness approaches lexicographic max-min fairness [19].

Given that the $\alpha$-fair objective is defined by equation (4), substitution into equation (2) results in a flow controller that will set its instantaneous rates based on the following equation:

$$
R_i(t) = \left( \frac{V}{U_i(t)} \right)^{1/\alpha} 
$$

(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 $\alpha = 5$.

### 6.2 Comparing backpressure and IFRC

We avoid using an arbitrarily high value to avoid queue drops.
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.

6.3 Static case

Figures 9(a) and 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 9(a) and 9(b), the rate vector presented by the backpressure stacks is lexicographically greater [5] 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 the achievable capacity [20]. 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 10.

Figure 11 shows the behavior of the PDQ stack and IFRC, under a dynamic flow scenario for 20 and 40 node topologies, with a fixed V.

![Figure 11: Behavior of PDQ and IFRC under dynamic flow scenario for 20 and 40 node topologies with a fixed V.](image)

Figure 12: Goodput performance of IFRC, PDQ and MDQ MAC, under a dynamic flow scenario, with fixed V. With fixed V it can be seen that the 4 flows {1, 7, 11, 13} for the 20-node case, and {20, 21, 30, 31} for the 40-node case) under-perform while running PDQ/MDQ, since IFRC shows that these flows can achieve a higher rate.

6.4 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 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 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 5. The maximum rate a node can achieve in the 20-node topology is therefore $\sqrt{610} = 3.66$ pkt/s. However, as can be seen from Figure 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 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 the backpressure-based rate control protocol 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 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 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 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. **CONCLUSION AND FUTURE WORK**

We undertook the first exhaustive empirical evaluation of backpressure based protocols in wireless sensor networks. We have shown that in this setting, a backpressure-based stack can be implemented with a simple scheme such as PDQ, with any need for modifications to the MAC layer. We also show that automatic parameterization is necessary for backpressure protocols 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. [3], 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 (and better than DQWM MAC) due to the small packet sizes that exist in wireless sensor networks. This condition should hold true for the
any-to-any case as well. 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.

Since one of the objectives of this work was to justify our claims that automatic parameter adaption of V is required in a dynamic flow setting, while performing the experiments with the adaptive V the value of V corresponding to a given number of flows was hard-coded into the protocol (Figures 12 and 13). In practice we will need to have online algorithms that can probe the network to estimate the appropriate V at which the network can operate. This will be the focus of our future work.

8. REFERENCES

[27] A. Warrier, L. Le, and I. Rhee. Cross-layer