Investigating Backpressure based Rate Control Protocols for Wireless Sensor Networks

Avinash Sridharan, Scott Moeller and Bhaskar Krishnamachari
Ming Hsieh Dept. of Electrical Engineering
University of Southern California, Los Angeles, CA 90089, USA
{asridhar, smoeller, bkrishna}@usc.edu
USC CENG Technical Report CENG-2008-7

***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 backpressure mechanisms for multi-hop wireless networks, in particular the first such study in the context of wireless sensor networks. Our evaluation yields two key insights into the design of such protocols. First, for wireless sensor networks, we show that a simple backpressure scheduling policy which allows nodes to transmit so long as they have a positive queue differential (irrespective of its size) gives performance comparable to more sophisticated heuristics. This result implies that, contrary to previous proposals, backpressure protocols can be implemented for wireless sensor networks 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.**

I. Introduction

The use of stochastic optimization techniques have resulted in backpressure based rate control mechanisms for wireless networks that show great promise ([14], [19], [22], [24]). At the core of these backpressure based algorithms is the scheduling policy proposed by Tassiulas [21] that resolves contention between nodes by scheduling the node with largest product of queue differential (between a node and its parent) and transmission rate. For a TDMA system, this scheduling policy is known to be throughput optimal [21]. As a result of this scheduling policy, the queues at a node are an indication of the congestion caused by flows originating at that node, as well as the path quality to the destination.

The techniques for designing rate control mechanisms over a backpressure scheduling policy were first introduced by Stolyar [20] and Neely *et al* [11], under the assumption of a TDMA system. Using these analytical techniques, it is possible to design flow controllers and routing modules that determine achievable flow rates and make next hop forwarding decisions. These modules use local queue sizes and one-hop queue differentials to optimize a specific concave rate based utility function. Cross layer solutions presented by these techniques had been restricted to the realm of theory because the backpressure scheduling policy is NP-hard. It is only recently that attempts ([14], [22]) have been made to use these techniques to design backpressure based protocols for widely prevalent CSMA-based wireless networks, employing sub-optimal heuristics.

Although existing proposals ([14], [22]) highlight the potential of using backpressure based techniques, we believe there are two key questions that are not addressed by these proposals. First, the optimality of the backpressure scheduling policy, proposed by Tassiulas [21], was proven under the assumption of a *maximum* match schedule. Since CSMA attempts to achieve a *maximal* match schedule, the value of complicated heuristics that approximate optimal backpressure scheduling over CSMA is unclear. Second, backpressure protocols rely on a single parameter to present a tradeoff between queue size and performance [22]. For these protocols, performance improves lograthimically with increase in queue size [11]. Given finite queues, and drastic reduction in per flow rate with increase in number of flows, it remains unclear whether good performance can be achieved under fixed parameter settings for dynamic traffic scenarios.

We present in this work a comprehensive experimental evaluation of backpressure based rate control protocols, the first-ever for wireless sensor networks. Our contributions in this work are two-fold: first, by performing a comparative evaluation of different heuristics for backpressure scheduling over a CSMA MAC, we show that in a sensor network setting a simple scheduling policy that allows node transmissions, when nodes have positive differential, performs as well as some of the proposed heuristics ([22], [24]) that prioritize node transmissions by modifying window sizes based on queue differentials. The key implication of this finding is that, in sensor networks, no modifications to the MAC are required in order to implement backpressure based protocols. Second, through a comparative evaluation of backpressure protocols against protocols optimized for wireless networks, we demonstrate that there exists no single parameter value that can guarantee optimal performance to backpressure based rate control protocols for a given topology. Further, we show that the optimal parameter setting is a function of the number of flows active in the network, and automatic parameter adaptation is therefore required for backpressure based protocols to perform well in a dynamic flow scenario.

Although we believe these questions are relevant in the context of any multi-hop CSMA based wireless network, our work is carried out in a low rate wireless sensor network setting. Our motivations for choosing sensor networks as a platform are as follows. The heuristics rely on modifying CSMA window sizes in an attempt to approximate backpressure scheduling over a CSMA MAC. In sensor networks since packet sizes are quite small (~ 40 bytes), the transmission time is usually smaller than the contention 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 contention window is usually 2.5 ms). Since multiple transmissions can take place within a single window size, increasing window size (as these heuristics do) might result in unnecessary loss of throughput, making the first question particularly relevant in this setting. Further, since sensor networks have allowed for clean slate design of protocols, there already exist rate control protocols in WSN that have been optimized for a multi-hop wireless network (IFRC [15]). The WSN setting therefore presents us with a good evaluation benchmark for answering questions regarding parametric dependence of backpressure protocol performance.

The paper is organized as follows; in section II, we present our related work. In section III, we present a software architecture that captures the general design of a backpressure based rate control stack. In section IV, we present the implementation details of heuristics that have been proposed for implementing backpressure scheduling over a CSMA stack. In section V, 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 VI, we present an evaluation of the backpressure based rate control protocols against IFRC [15] in order to understand the parameter dependence of backpressure protocols. In section VII, we present a summary of our results and future directions for this work.

II. Related Work

Tassiulas *et al.* [21] used Lyapunov Drift techniques to demonstrate the existence of a backpressure scheduling policy whose network capacity region is a superset of the capacity region for all alternative scheduling policies. Their work assumes TDMA synchronized operation, with a centralized scheduler. Two independent branches of analysis have emerged, which have justified and extended backpressure algorithms, since the work presented by Tassiulas *et al* [21]. Neely *et al* ([9], [10], [11]) built on the Lyapunov Drift framework and extended it to support joint utility and throughput optimization. These works introduce V, a constant which prioritizes the utility optimization over queue backlog minimization. The resulting algorithm approaches the optimal utility logarithmically with increasing V, while the bound on system queue backlog grows linearly with increasing V.

A second analytical approach has been developed by Alexander Stolyar [20]. He leveraged the primal-dual gradient descent techniques in defining the Greedy Primal-Dual Algorithm (GPD). Interests in primal-dual algorithms for congestion control and flow scheduling were initially sparked by Kelly, Maulloo and Tan [7]. Within the analytical framework proposed by Stolyar, $\beta > 0$ is a small constant parameter which tunes the utility optimization. As $\beta \rightarrow 0$ the system performs arbitrarily close to utility optimal, at a cost of increased queue sizes.

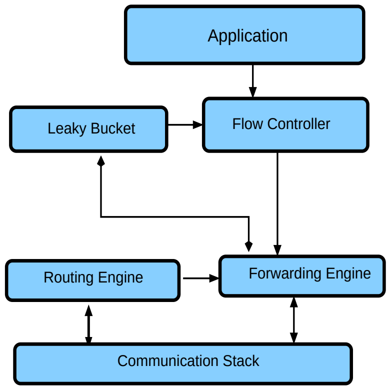


Fig. 1. The software architecture for a backpressure based rate control stack.

In an attempt to implement backpressure scheduling on CSMA wireless networks, Warrier *et al* [24] and Umut *et al.* [22] 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. Umut *et al.* [22] also proposes design of flow controllers on top of these schedulers based on the technique proposed by Stolyar [20]. The work by Bozidar *et al* [14] develops a multipath routing and rate control protocol, that can be integrated with TCP over 802.11, using backpressure techniques. They use a naive backpressure scheduler that allow transmissions as long as the queue differential is greater than a threshold. A drawback of all these proposals is that it is unclear 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 [2]. We believe this lack of evaluation with protocols that have been optimized over wireless, hides the parametric dependence of backpressure protocol performance.

In the context of wireless sensor networks, Sridharan *et al* [19] have designed flow controllers for a CSMA based sensor network, using the framework proposed by Neely *et al.* [11]. In terms of rate control, there have been several proposals in wireless sensor networks ([4], [6], [12], [15], [17], [18], [23], [25]). Most of these protocols have assumed a clean slate design and follow a router centric, explicit congestion notification approach. Of these protocols, IFRC [15] and WRCP [18] are distributed protocols that attempt to achieve lexicographic max-min fairness in conjunction with congestion control. We choose IFRC [15] as a benchmark to demonstrate that backpressure protocol performance is dependant on parameter settings.

III. Backpressure based rate control stack

We start our investigation by presenting a software architecture, shown in figure 1, that captures the design of a generic backpressure based rate control stack. For tractability, 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. Although the results presented here are specific to a collection tree, we present logical arguments in section VII to show that they apply to a general setting.

A backpressure based rate control algorithm has two parts: a flow controller and a backpressure based scheduler. The functionality of the flow controller is implemented as part of the "Leaky Bucket" and "Flow Controller" blocks in figure 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 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 in turn uses the admission rate to control the rate at which tokens are generated. When a packet arrives from the application at the flow controller, it is injected into the forwarding engine only if a token is available from the leaky bucket.

The functionality of the backpressure based scheduler is to determine when to allow a node to transmit. This decision is based on its current queue differential with its parent. In figure 1, the backpressure scheduler is implemented as part of the "Forwarding Engine" and "Communication stack" blocks. 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 wether 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 backoff window sizes before injecting the packet.

We now describe the implementation of the flow controller and backpressure scheduler in further detail.

A. Flow controller design

The objective of the flow controller is to maximize $\sum_{\forall i} g(r_i)$, where r_i is the time average source rate and $g(r_i)$ is a concave utility function. In order to design such flow controllers we can use one of two techniques presented by Stolyar [20] and Neely *et al.* [11].

In the proposal presented by Sridharan *et al.* [19], the flow controller is designed using a technique proposed by Neely *et al.* [11]. In this design, at every time step t, the instantaneous rate $R_i(t)$ at which packets are admitted into the system is that which maximizes the following equation:

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

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

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

Here V is a constant that acts as a tuning parameter to effect a tradeoff between the forwarding queue size U_i and value of $g(r_i)$. 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 $g(r_i)$.

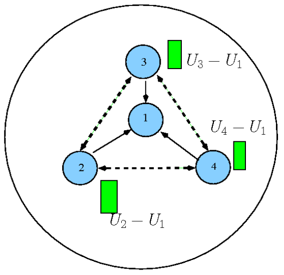


Fig. 2. Understanding maximum differential queue scheduling over a CSMA based wireless network.

It should be noted that flow controllers designed using the technique proposed by Stolyar [20] are very similar in structure to the one shown in equation 2. This is highlighted by the proposal from Umut *et al.* [22]. The only difference between the two designs is the parameter β, which has an inverse effect as compared to V. A small β implies better optimality and larger queues. A large β implies smaller queues but lower utility. We therefore chose a flow controller, based on equation 2, without affecting the generality of the results. We make this choice primarily due to our familiarity with the technique proposed by Neely [11], and code availability for the flow controller, based on equation 2, for a sensor network setting [19].

B. Scheduler design

Next we explain the optimal backpressure scheduling policy proposed by Tassiulas *et al.* [21]. 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 contending to transmit to the sink and link rate for all sources is assumed equal, the backpressure based scheduler will select the node with the largest queue differential. The optimal backpressure scheduler assumes that a TDMA MAC exists which will present it with a maximum match schedule.

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 contention window. Proposals ([22], [24]) therefore try to achieve prioritization of node transmission by changing the CSMA window size based on the current queue differential. We refer to these heuristics as *queue differential prioritization* techniques. Umut *et al.* [22] 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. For calculating weight nodes need to transmit their queue differential explicitly.

Warrier *et al.* [24] 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.

Given that the optimality of backpressure scheduling is proved for a maximum match schedule [21], and the heuristics presented above result in a maximal match schedule, the heuristics will be sub-optimal. A simpler sub-optimal approach, that results in a backpressure signal to the source, could be to allow the forwarding engine 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 a *pure backpressure* scheme. This scheme is similar to the one used by Bozidar *et al* in [14].

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

IV. Implementing backpressure scheduling over a CSMA based WSN

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 [22], and the pure backpressure scheme over the CC2420 CSMA MAC. In order to test differential queue prioritization, we chose to implement the scheme proposed by [22] and the not the scheme proposed by Warrier *et al* [24]. This is because the scheme proposed by Warrior *et al* requires a mapping between the queue differential and window sizes which itself is a heuristic. Given that the performance of the scheme proposed by [24] depends heavily on the choice of the mapping, it adds another dimension to the comparison which is hard to quantify. We first present a description of the CC2420 CSMA MAC over which the schemes will be implemented.

A. The CC2420 CSMA MAC

The CSMA-CA algorithm in CC2420 CSMA MAC operates on only two types of backoff windows, the initial backoff window W_i, and a congestion backoff window W_c. When a node injects a packet into the MAC layer, the MAC layer performs a random backoff between $[0, W_i]$. At the end of the initial backoff 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 backoff stage performing a random backoff 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 backoff slot duration is 32.25 microsecond resulting in a 10 ms (320×32.25) initial backoff window, and 2.58 ms (80×32.25) congestion backoff window.

B. Implementing differential queue prioritization: the MDQ MAC

The maximum differential queue prioritization technique (MDQ), proposed by Umut *et al.* [22], and described in section III-B, 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. 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 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. The MDQ proposed in [22] chooses to perform maximum weight calculation in a 2-hop neighborhood. We feel this might be too conservative a design choice. Second, if a node does not have the maximum weight in a neighborhood, it can modify both the initial backoff window (W_i) and the congestion backoff window (W_c), or just the congestion backoff window (W_c). The cost imposed on a node, when it does not have maximum weight in a neighborhood, is higher if W_i and W_c are both increased, as compared to increasing only W_c. Intuitively it seems changing W_i and W_c simultaneously is a conservative approach, since node transmissions should be prioritized only if contention happens. To verify this intuition, the performance when both initial and congestion backoff windows are modified is compared to an implementation that modifies only the congestion window.

To cater for the various combination of the above design choices, we have implement multiple version of the MDQ MAC, each MAC is titled MDQn-INITm or MDQn-CWm. The variable n represents wether the calculation is performed in a 1-hop neighborhood or a 2-hop neighborhood (hence $n = \{1, 2\}$). For MDQn-INITm, the max weight is calculated in a neighborhood of size n and if a node is not the maximum weight its initial and congestion backoff windows are increased by $m \cdot W_i$ and $m \cdot W_c$ respectively. For MDQn-CWm if a node is not the maximum weight only its congestion backoff window is increased by $m \cdot W_c$. We choose m to be either 2 or 4.

MDQ1-INITm and MDQ1-CWm calculates the maximum weight in a 1-hop neighborhood by maintaining a list of weights of its neighbors, and calculating the max between its own weight and this list. 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. MDQ2 uses the algorithm presented in [22] to calculate the maximum weight in a 2-hop

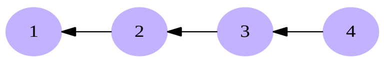


Fig. 3. The 4 node fully connected topology with linear routing.

neighborhood. For MDQ2-INITm and MDQ2-CWm, nodes transmit their 1-hop neighborhood max along with their own weights (this requires an extra field to be added to the MAC header). The 1-hop neighborhood maximum is calculated as described above, the 2-hop neighborhood maximum at a node is the maximum amongst the 1-hop neighborhood max, overheard from all neighboring nodes.

C. Implementing pure backpressure: the PB MAC

The pure backpressure scheme is trivial to implement. In the pure backpressure (PB) 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 pure backpressure scheme thus does not perform any prioritization of node transmissions, and hence does not require to modify the MAC window sizes.

V. Evaluating differential queue prioritization in a CSMA based WSN

In order to evaluate the performance of various MDQ MAC schemes presented in section IV against the PB MAC, we implement a backpressure based rate control stack with a log utility flow controller. The flow controller is run on top of different MAC schemes to present a comparative evaluation. The different version of the MDQ MAC, labeled either MDQn-INITm or MDQn-CWm, have been described in section IV.

We run each of these stacks on the 3 different topologies shown in figures 3, 4, and 5. The 4 node topology of figure 3 is a fully connected topology on which a linear routing tree has been imposed. The 20 and 40 node topologies in figures 4 and 5 are multi-hop topologies where nodes are not fully connected. They are formed using the USC Tutornet testbed [8], a 100 node WSN research testbed that spans two floors. Figure 6 indicates the connectivity levels for both 20 and 40 node topologies under transmit power levels $\{5, 10\}$. The metrics used to compare the different stacks are the total log utility and average total queue size. We test each topology over varying power levels. We first present the implementation details of the log utility flow controller. where

A. Log utility flow controller

Design of the log utility flow controller follows the description presented in section III-A. The log utility flow controller tries to maximize the global utility function $\sum_{\forall i} g(r_i)$, where $g(r_i) = \log(r_i)$. Therefore, by equation 2, the instantaneous rate $R_i(t)$ is:

$$R_i(t) = \frac{V}{2U_i(t)} \tag{3}$$

Where V is a constant parameter. We show how this parameter is chosen in the next section. Our choice of the log utility flow

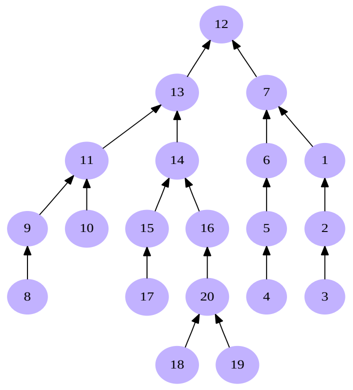


Fig. 4. The 20 node routing tree

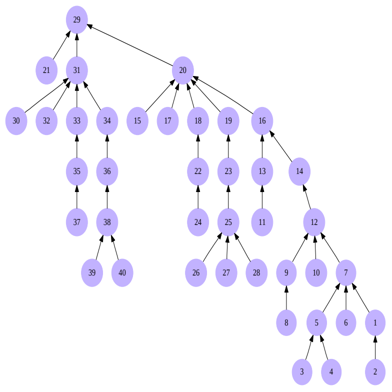


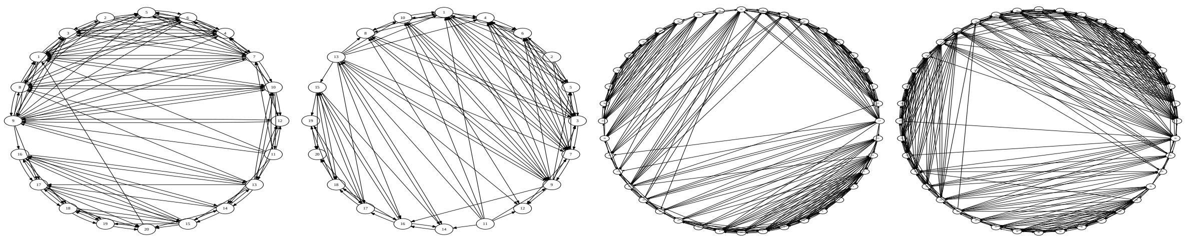
Fig. 5. The 40 node routing tree

controller for performing this evaluation is motivated by the fact that in wired networks, maximizing the total log utility amounts to achieving proportional fairness [7]. Further, in a wireless setting [16] shows that log utility presents the best trade-off between fairness and efficiency.

B. Parameter selection

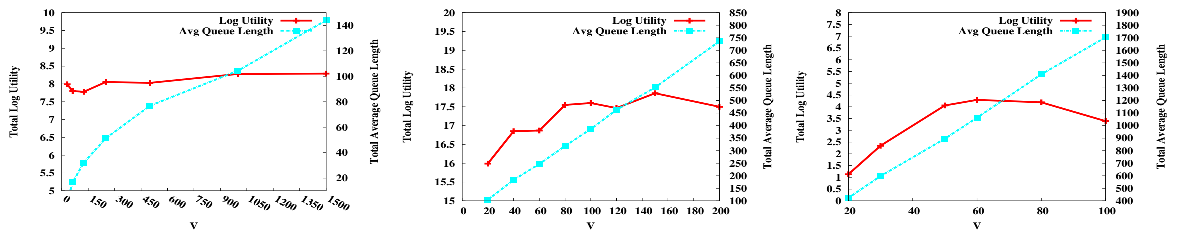
As described in section III, the fixed parameter V presents a trade-off between the utility achieved and the average system queue size. In order to tune V optimally for the PB MAC stack, we plotted the total log utility and average total queue size for the three different topologies shown in figure 7. For each topology, the system utility increases with V and saturates beyond a certain value of V. 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 value of 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 value of V the required queue sizes cannot be supported. This results in packet drops, lowering total utility. Using figure 7 we select a value of $V = 1500$ for the 4 node topology, $V = 150$ for the 20 node topology, and $V = 60$ for the 40 node topology.

Although the plots in figure 7 were obtained using the PB 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



(a) 20 node (Power Level =5) (b) 20 node (Power Level =10) (c) 40 node (Power Level =5) (d) 40 node (Power Level =10)

Fig. 6. Connectivity for the 20 and 40 node topologies. Connected nodes indicate PRR of at least 80%.



(a) 4 node (b) 20 node (c) 40 node

Fig. 7. Selection of the parameter V for different topologies.

monotonically with V [5] 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 for that stack in order to fairly compare performance.

C. Comparing PB with MDQ MAC

Figures 8 and 9 present 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 three 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. For the 4 node topology we perform experiments only at single power level since the nodes are in close proximity. Even at this low power level they are fully connected.

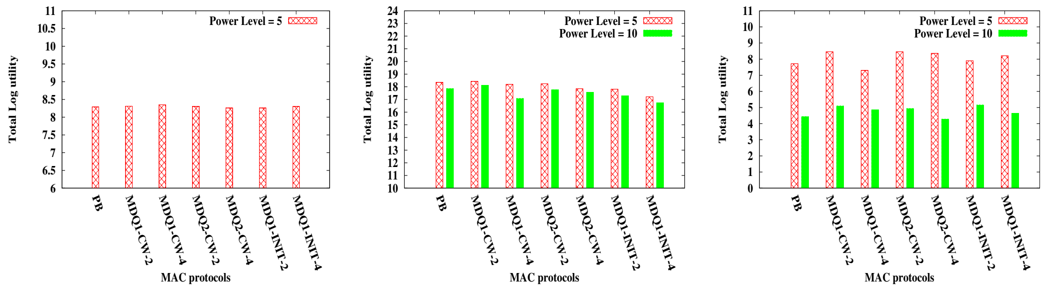
The log utility performance in figure 8 shows interesting behavior across topologies. The total log utility increases from the 4 node to the 20 node topology, and decreases from the 20 node to the 40 node topology. Additionally, for 40 nodes, log utility decreases when the power level is increased. The reason for the increase in log utility for 20 nodes is that the rates for all sources remain greater than 1, and because the number of flows increases the sum log utility increases as compared to the 4 node topology. 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 6. This results in reduced available capacity and hence leads to smaller total log utility.

For the 4 node topology, Figure 8(a) indicates that the log utility performance of all stacks are equivalent. In terms of the average queue sizes, the MDQ1-CW2 stack out performs the PB stack by only 2%.

In the 20 node topology, the PB stack performs similar to the MDQ1-CW2 MAC and outperform all the other stacks, in terms of total log utility as well average total queue size. This is true for both power levels. For the 40 node topology, however, the MDQ1-CW2 MAC and the MDQ2-CW2 MAC outperform the PB MAC by a small margin. In terms of log utility the MDQ1-CW2 MAC and MDQ2-CW2 outperform the PB MAC by ~ 0.5 and total queue sizes are reduced by ~ 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 $\frac{\prod^{MDQ1-CW2} r_i}{\prod^{PB} r_i} = e^{0.5} \sim= 1.5$. 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 is to 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 $\frac{10}{40} = 0.25$ packets under PB MAC, as compared to the MDQ1-CW2 MAC.

Apart from a comparative evaluation of PB against MDQ MAC, another insight that can be learned from these results is that across all topologies and across all power levels, the MDQn-CWm always outperforms the MDQn-INITm MAC. This implies that our arguments against modification of the initial backoff window, in this specific setting of small packet sizes, is valid. Further, the MDQ1-CWm outperforms the MDQ2-CWm, implying that performing maximum weight calculations in a 2-hop neighborhood is unnecessary.

The key implication of the results presented in figures 8 and 9 is that the pure backpressure scheme performs comparable to the various MDQ MAC scheme across different size



(a) 4 node, V = 1500 (b) 20 node, V = 150 (c) 40 node, V = 60

Fig. 8. Log utility performance of different MAC protocols across different topology sizes and different power levels.

topologies, and for different power levels. From the description presented in section IV, it is clear that the complexity introduced by the MDQ 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 backoff windows, backpressure based protocols can be implemented with similar performance over existing CSMA MAC by simply implementing pure backpressure on top of them.

VI. Understanding backpressure protocol performance in dynamic flow scenario

Our second goal is to understand the relationship between backpressure protocol performance and its fixed parameter setting, specifically in a dynamic flow scenario. For the backpressure rate control stack, we choose two versions: one running the PB MAC and one running the MDQ1-INIT4 stack. We choose these two MAC implementation because they are two extremes of the variants that we evaluated in section V. As was seen in section III-A, the only parameter that the protocol performance depends on is V. In order to gage 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 [15]).

IFRC [15] is an additive increase multiplicative decrease protocol that attempts to achieve lexicographic max-min fairness [3] over a collection tree. IFRC uses a preset queue threshold to detect congestion and send explicit congestion notification.

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 flow are active for the entire duration of the experiment. We consider a dynamic flow scenario as one in which only a subset of flows are active for the entire duration while remaining flows join the network at pre-specified intervals.

As IFRC aims to achieve lexicographic max-min fairness, a valid comparison cannot be achieved using the log utility flow controller described in section V. Instead we design a new flow controller using the notion of α-fairness. We describe the design of the α-fair controller before presenting our comparative results.

A. *α-fair controller*

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

$$\sum_{\forall i} \frac{r_i^{1-\alpha}}{1-\alpha} \tag{4}$$

Here, α is a constant greater than 1. Theoretically, it has been shown that when $\alpha \to \infty$, α-fairness approaches lexicographic max-min fairness [13].

Given that the α-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:

$$R_i(t) = \sqrt[\alpha]{\frac{V}{U_i(t)}} \tag{5}$$

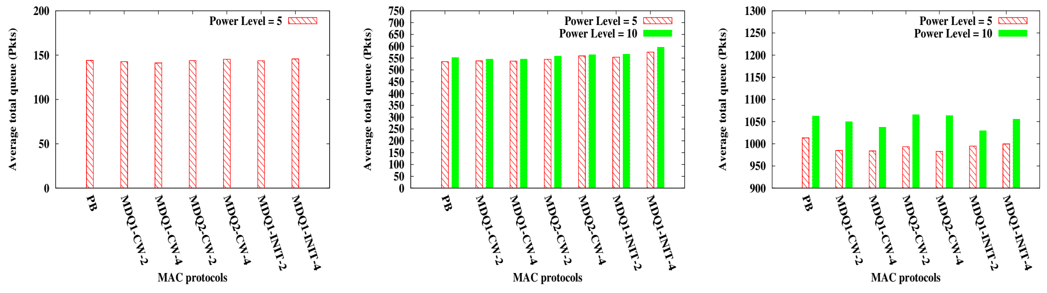
In order to achieve lexicographic max-min fairness we want α to be large. We are able to achieve results comparable to IFRC for our specific sensor network setting with $\alpha = 8$.

B. *Comparing backpressure and IFRC*

The queue threshold we use for IFRC is 20 packets. The parameters for 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 $V = 30000$ for the 20 node scenario and $V = 10$ for 40 nodes. This resulted in an average per-node queue size of approximately 20 packets under the backpressure stacks.

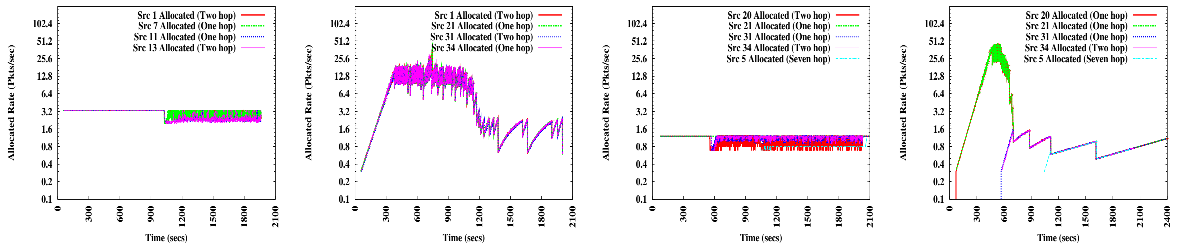
Figures 11(a) and 11(b) show the static flow goodput performance of the PB stack, MDQ1-INIT4 stack, and IFRC over the 20 and 40 node topologies. We present the results for the static scenario to justify our flow controller implementation. As can be seen in figure 11(a) and figure 11(b) the rate vector presented by the backpressure stacks is lexicographically greater [3] 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.

We now evaluate the dynamic flow setting. To generate a dynamic scenario on the 20 and 40 node topologies we use the following flow activation strategies. In the 20 node topology,



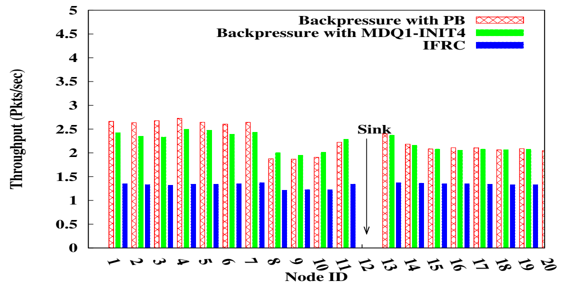
(a) 4 node, V = 1500 (b) 20 node, V = 150 (c) 40 node, V = 60

Fig. 9. Average total queue length for different MAC protocols across different topology sizes and different power.

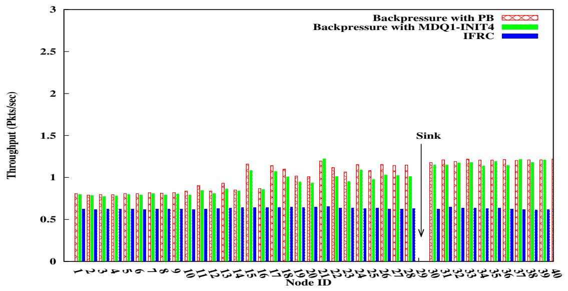


(a) 20 node dynamic (PB) (b) 20 node dynamic (IFRC) (c) 40 node dynamic (PB) (d) 40 node dynamic (IFRC)

Fig. 10. Behavior of PB and IFRC under dynamic flow scenario for 20 and 40 node topologies.



(a) 20 node (static)



(b) 40 node (static)

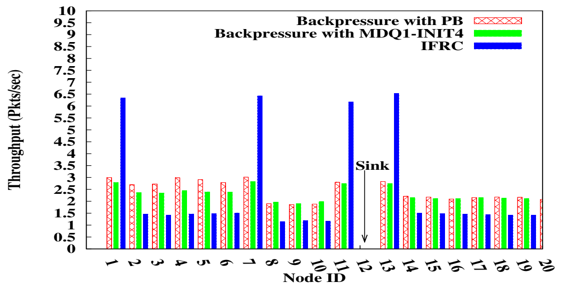
Fig. 11. Comparing goodput performance of IFRC and backpressure based stack with PB and MDQ MAC on 20 and 40 node topologies, under a static flow scenario.

nodes $\{1, 7, 13, 11\}$ are active for the complete duration of the experiment while all other nodes are activated at 1200 seconds into the experiment. For the 40 node topology, nodes $\{20, 21\}$ are active for the entire duration of the experiment while all other flows are activated at 800 seconds into the experiment.

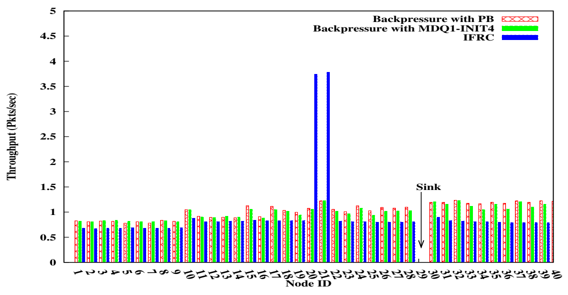
Figure 10 shows the behavior of the PB stack and IFRC in a dynamic setting. Note that the y-axis in each of these graphs is in log scale. For both topologies, it can be seen that when a few flows are operational in the network, the goodput given to these flows is much higher in the case of IFRC as compared to the PB stack. This can be seen between $0 - 1200$ seconds for the 20 node topology, and $0 - 600$ seconds for the 40 node topology. When all flows become active (at 1200 seconds for 20 node, and 600 seconds for 40 nodes) the scenario becomes the same as the static case, and as seen before PB outperforms IFRC. Due to space constraints, MDQ1-INIT4 graphs are not presented. But as seen from the goodput in figure 12, the performance of MDQ1-INIT4 is similar to PB.

The above behavior can be explained by our fixed V selection. For the 20 node topology $V = 30000$. 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 $\sqrt[8]{30000} = 3.6277$ pkts/sec. However, as can be seen from figure 10(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.

The experiments presented in this section clearly show the underperformance of backpressure protocols under constant parameter settings. Our motivation for presenting these results was to highlight an explicit need for design of automatic parameter adaption in backpressure protocols for wireless networks. Though these results are specific to a sensor network setting, the variation in available capacity and restrictions on V due to finite queue sizes are realities that will be common



(a) 20 node (dynamic)



(b) 40 node (dynamic)

Fig. 12. Comparing goodput performance of IFRC and backpressure based stack with PB and MDQ MAC on 20 and 40 node topologies, under a dynamic flow scenario.

across all wireless networks.

VII. 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 pure backpressure approach performs comparably to schemes that attempt differential queue prioritization as a means for approximating optimal backpressure scheduling. This implies that backpressure protocols can be developed on existing CSMA MAC without modifications. 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 Umut *et al.* [22], support for any-to-any traffic is possible through the addition of per destination queues. For the single destination case, we reason that PB 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.

The focus of our future work will be to develop algorithms that can achieve automatic parametrization for backpressure protocols, in a dynamic flow setting.

References

[1] *http://www.moteiv.com*.

[2] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz. Improving TCP/IP performance over wireless networks. *Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 2–11, 1995.

[3] Bertsekas and Galagher. Data networks. *Prentice Hall*.

[4] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. *ACM Sensys*, 2004.

[5] L. Georgiadis, M.J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking, Vol. 1, no. 1, pp. 1-144*, 2006.

[6] B. Hull, K. Jamieson, and H. Balakrishnan. Techniques for mitigating congestion in sensor networks. *ACM Sensys*, 2004.

[7] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[8] Embedded Networks Laboratory. *http://testbed.usc.edu*, 2007.

[9] M.J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, 2003.

[10] M.J. Neely. Energy optimal control for time varying wireless networks. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 1, 2005.

[11] M.J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proceedings of IEEE INFOCOM*, 2005.

[12] J. Paek and R. Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. *Sensys*, 2007.

[13] B. Radunovi and J.Y. Le Boudec. Rate Performance Objectives of Multihop Wireless Networks. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, pages 334–349, 2004.

[14] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: Balancing TCP over multiple paths in wireless mesh network. *ACM Mobicom*, 2008.

[15] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. *In Proceedings of ACM SIGCOMM Symposium on Network Architectures*, 2006.

[16] P. Room and S. Cart. Rate performance objectives of multihop wireless networks.

[17] Y. Sankarasubramaniam, O.B. Akan, and I.F. Akyildiz. ESRT: Event-to-sink reliable transport in wireless sensor networks. *ACM MobiHoc*, 3:177–188, 2003.

[18] A. Sridharan and B. Krishnamachari. Achieving fast convergence for max-min fair rate allocation in wireless sensor networks,. 2008.

[19] A. Sridharan, S. Moeller, and B. Krishnamachari. Making distributed rate control using lyapunov drifts a reality in wireless networks. *6th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2008.

[20] A.L. Stolyar. Maximizing queueing network utility subject to stability: greedy primal-dual algorithm. *Queueing Systems*, 50(4):401–457, 2005.

[21] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control, 37(12)*, 1992.

[22] A. Umut, M. Andrews, P. Gupta, J. Hobby, I. Sanjee, and A. Stolyar. Joint scheduling and congestion control in mobile ad-hoc networks. *IEEE Infocom*, 2008.

[23] C.Y. Wan, S.B. Eisenman, and A.T. Campbell. CODA: Congestion detection and avoidance in sensor networks. *The First ACM Conference on Embedded Networked Sensor Systems*, 2003.

[24] A. Warrier, L. Le, and I. Rhee. Cross-layer optimization made practical. *Broadnets, Invited Paper*, 2007.

[25] A. Woo and D.E. Culler. A transmission control scheme for media access in sensor networks. *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, 2001.