USC Computer Engineering Technical Report CENG-2009-10

Backpressure Routing Made Practical

Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, Omprakash Gnawali
Department of Electrical Engineering
University of Southern California, Los Angeles, 90089
{smoeller,asridhar,bkrishna,gnawali}@usc.edu

The current design methodology for data collection protocols in wireless sensor networks relies on the proactive construction and maintenance of quasi-static shortest path trees for routing. We consider an alternative highly-agile approach called backpressure routing, in which forwarding decisions are made on a per-packet basis using backpressure link weights that incorporate local queue state and link quality information. In theory, according to recent literature on cross-layer stochastic network optimization, such a dynamic routing approach can provide near-optimal utilization of the available bandwidth. However, it has not been implemented on practical systems to date due to concerns about packet looping, the effect of link losses, and large delays. Addressing these concerns, we present the Backpressure Collection Protocol (BCP), the first ever implementation of dynamic backpressure routing in wireless sensor networks. Through experiments on a 40-node wireless sensor network testbed, we demonstrate that incorporating transmission count minimization in the backpressure weight eliminates packet looping and provides excellent throughput performance. We show that BCP provides more than 50% improvements in throughput compared to a state of the art tree-based collection protocol. We also show that the average end-to-end packet delays in BCP can be drastically reduced (75% under high load, 98% under low load) by using LIFO queues.

I. Introduction

As wireless sensor networks mature from concepts and simple demonstrations to real-world deployments, there has been a push to identify and develop key networking building blocks in a more organized and coherent fashion. One such fundamental building block that has been identified at the network layer is *Collection*, which allows for data from multiple sources to be delivered to a common sink. State-of-the-art implemented protocols for collection are based on quasi-static minimum cost trees with suitably defined link metrics [9]. Due to the limited radio link rates, high density of deployment, and multi-hop operation, bandwidth is a scarce resource in wireless sensor networks, and recent studies such as [4] have suggested that it is essential to improve collection throughput as much as possible.

We explore in this work an exciting alternative approach — *dynamic backpressure routing* — that allows for load-balanced re-routing around hot-spots on a per-packet basis to substantially enhance the throughput efficiency of collection. This technique is based on elegant theoretical results pertaining to differential queue backpressure scheduling and cross-layer stochastic optimization ([10], [14]). The crux of this approach lies in calculating link weights that take into account the difference of queue size across a link, as well as link quality information, and then prioritizing forwarding over high-weight links. In theory, backpressure mechanisms promise throughput-optimal performance and elegant cross-layer solutions for integrating medium access, routing, and rate control.

Despite the theoretical promise of backpressure techniques, to date it has not been implemented in practice at the routing layer[1] due to several challenges. First, if the link weights are not carefully defined, backpressure routing can suffer from either excessively high hop-counts (packet looping) or, at the other extreme, over-emphasize low hop counts, resulting in wasted transmissions over lossy links. Second, due to relatively high queue sizes that must be maintained to provide a gradient for data flow even at low loads, backpressure routing can suffer from inordinately large delays. Finally, while most of the theoretical work on backpressure scheduling provide for centralized TDMA solutions, in practice a distributed asynchronous routing solution is needed.

In this work, we take the first steps towards addressing these problems in order to allow backpressure routing to realize their promise in practical environments.

We present in this paper the Backpressure Collection Protocol (BCP), a low-overhead dynamic backpressure routing protocol at the network layer implemented on TinyOS 2.x, a widely used wireless sensor network operating system and protocol stack. We evaluate it through real experiments on a 40-mote testbed, where we compare BCP's performance with the Collection Tree Protocol (CTP)[9], a state of the art routing protocol distributed with TinyOS 2.x. We find that BCP outperforms CTP in terms of maximal achievable network goodput by more than 50%. BCP incorporates a novel link-specific-threshold to minimize the expected number of transmissions (ETX) per delivered packet, essential for energy-efficient and robust performance over real-world networks with lossy links. We show that by using LIFO queueing instead of

[1]There have been several implementations of backpressure ideas at the MAC and transport layers, as we shall discuss when presenting related work in section V.

FIFO, the delays associated with backpressure routing can be reduced dramatically, by more than 98% at low data rates and by 75% at high data rates without affecting the achievable goodput.

II. Backpressure Explained

Unlike traditional routing mechanisms for wired and wireless networks, backpressure routing does not perform any explicit path computation from source to destination. Instead, the routing and forwarding decision is made independently for each packet by computing a backpressure weight on each outgoing link that uses only localized queue and link state information. A generic form for this backpressure weight $w_{i,j}$ calculated by a node i for a given neighbor j is the following:

$$w_{i,j} = (\Delta Q_{i,j} - \theta_{i,j}) \cdot R_{i,j} \tag{1}$$

Here, $\Delta Q_{ij} = Q_i - Q_j$ is the queue differential, with Q_i and Q_j representing the number of packets in the queues of nodes i and j respectively[2], $R_{i,j}$ is the link rate, and $\theta_{i,j}$ a link-specific threshold that can be used to optimize a given objective such as minimizing hop-count or the number of transmissions incurred to deliver each packet. For each packet, node i computes the backpressure weight $w_{i,j}$ for all its neighbors, and uses it as the basis for making routing (who to try and send the packet to) and forwarding (whether to send the packet) decisions as follows. **Routing decision:** Node i identifies the link (i, j^*) with the highest value of the backpressure weight as the next hop for the packet. **Forwarding decision:** if $w_{i,j^*} > 0$, the packet is forwarded (i.e. sent to the link layer for transmission to the designated neighbor), else the packet is held back for some time τ before the metric is recomputed.

In describing this simple routing and forwarding mechanism to colleagues unfamiliar with backpressure techniques, we have found that a common initial reaction is surprise that this simple forwarding strategy that has neither an explicit path computation nor an explicit reference to the destination, should work at all.

We illustrate the functioning of backpressure routing with a simple example. Figure 1 shows a linear topology with four nodes labelled 3, 2, 1, and S (for sink) respectively. For simplicity, assume that $\theta_{i,j} = 1, R_{i,j} = 1$. For the following explanation, we can look at either the top or the bottom row (we shall discuss later the difference between these cases.) In steady state, there is a natural queue backpressure gradient sloping downwards to the sink [3]. Each node has just one packet more than its neighbor to the right but is unable to forward because it does not strictly exceed the threshold of 1. The injection of new packets into nodes 1 and 2, shown in step B, causes the thresholds to be exceeded. Node 1 then starts sending packets to the sink, while node 2 initially forwards a packet backwards to node 3 (after step B), then halts (after step C), then reverses to start sending packets to node 1 as that node's packets are drained out by the sink. Eventually six packets (corresponding to the number of new arrivals) are sent to the destination, and the system returns to the steady state gradient.

[2]As we are focusing on data collection settings in wireless sensor networks where packets are all intended for a common sink, it suffices to maintain a single queue at each node. For more complex multi-commodity settings, backpressure routing may require the maintenance of multiple queues.

[3]Backpressure routing requires a gradient to exist before packets can begin to be forwarded, resulting in a small startup time, and the possible sacrifice of a small number of "trapped" packets. Both of these are negligible concerns for even moderately long flows.

The main benefit of using backpressure routing is its throughput performance. While it cannot be seen from the simple one-dimensional illustration above, backpressure routing makes very efficient use of the available bandwidth by dynamically re-routing packets away from hot-spots. It was originally shown by Tassiulas and Ephremides [14] that a centrally scheduled version of backpressure routing is throughput optimal. We show through experimental measurements in this work that with our distributed scheme we are able to obtain a dramatic 50% improvement in total throughput compared to shortest path routing.

The performance of backpressure routing is also highly sensitive to the choice of the backpressure weight. A weight that looks only at queue differentials and imposes no positive threshold can result in excessively long paths (looping). On the other hand, as we shall show, a backpressure weight that emphasizes hop count minimization results in wasted transmissions due to selection of high-loss links. In this work, we use a novel backpressure weight that is theoretically derived to minimize expected transmission counts and show that this provides highly efficient performance.

There is a major shortcoming of backpressure routing when performed in conjunction with traditional FIFO queuing — extremely large delays. In the top row of figure 1, which illustrates forwarding with first-in-first-out (FIFO) queues, we see how some of the incoming packets marked in black settle down to help form the backpressure gradient; these packets can experience inordinately high delays until they are dislodged by new arrivals. And (somewhat counter-intuitively, perhaps, if one does not see this illustration) these delays are highest when the traffic load in the network is low. As we shall show through our experimental evaluations, average packet delays for backpressure routing with FIFO queues can be as much as three orders of magnitude worse than with shortest path routing at low rates.

A key contribution of our work is to show for the first time that this problem can be substantially alleviated by using the last-in-first-out (LIFO) discipline instead. As illustrated in the second row of Figure 1, with a LIFO discipline, while the packets that form the backpressure gradient remain "trapped", the new packet arrivals into the network are all rapidly delivered to the sink. Thus, by sacrificing a small number of packets at the very beginning to create the backpressure, the delay performance can be greatly improved. We will empirically show that LIFO queues provide up to two orders of magnitude improvement in delay performance of backpressure routing for low rate settings compared to FIFO.

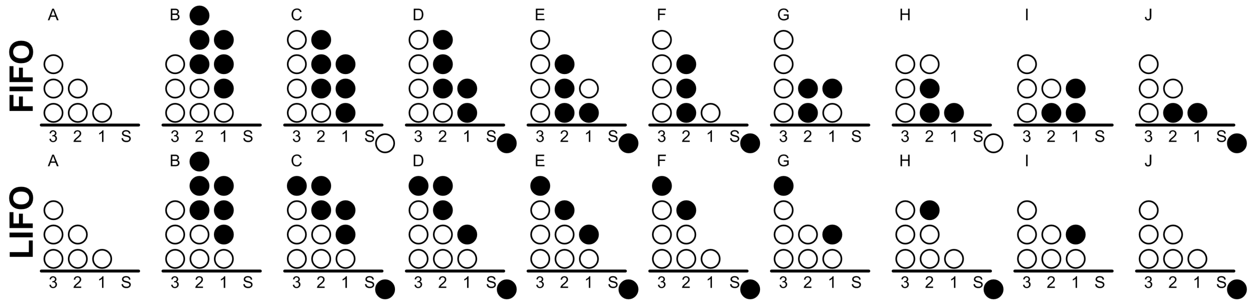

Fig. 1: An intuitive example of backpressure routing on a four-node line network. Three packets (black circles) are injected at nodes 1 and 2 at time B, intended for the destination sink S.

III. BCP Implementation

We have developed the Backpressure Collection Protocol (BCP), the first ever real-system implementation of a dynamic backpressure routing mechanism. BCP is implemented on TinyOS 2.x, and has been tested on the IEEE 802.15.4-based Tmote Sky platform. BCP's code footprint is about 23 KB including our test application.

One design decision that needs to be made in a real system implementation is how to choose the link-specific threshold $\theta_{i,j}$ used to make the forwarding decision. We consider the following two thresholds: $\theta_{i,j}^{HOP} = V$, and $\theta_{i,j}^{ETX} = V \cdot ETX_{i,j}$, where V is an empirically determined parameter that provides a tradeoff between Hop/ETX minimization and delay. These thresholds are not mere heuristics but are mathematically derived using the Stochastic Lyapunov-Drift Optimization framework developed by Neely [10]. Using $\theta_{i,j}^{HOP}$ minimizes the total expected number of hops per packet, while $\theta_{i,j}^{ETX}$ minimizes the total expected number of transmissions per packet [4].

Our emphasis in this work is on the novel transmission-count minimizing backpressure weight that uses $\theta_{i,j}^{ETX}$. We will therefore refer to our implementation of backpressure routing with this weight as BCP. We implemented the hop count minimizing backpressure weight that uses $\theta_{i,j}^{HOP}$ primarily to provide a reference comparison. Since this weight is similar to what has been previously proposed in the literature as Volcano routing [8] (although that work did not include a link rate product term), we refer to this variant as Volcano. We will show in Section IV that while Volcano provides low hop counts, this comes at the expense of high transmission counts and packet losses due to its use of high-loss links (similar to what was observed in [6]). The use of ETX minimization in the context of backpressure routing is a novel contribution of this work, and as we shall show, is essential for low losses over real-world wireless networks with lossy links.

[4]We omit the formal details of the mathematical derivation due to space constraints, but essentially, what can be shown is that if link weights $w_{i,j}$ are calculated as per equation (1) with $\theta_{i,j}$ set to $\theta_{i,j}^{ETX}$ (respectively, $\theta_{i,j}^{HOP}$), and these weights are used as the basis for a TDMA scheduling where at each time step non-interfering links that provide maximum-weight matching are chosen, then the resulting dynamic backpressure routing algorithm is a stochastic solver for the following wireless network optimization problem: minimize the expected number of transmissions per packet (respectively, expected number of hops per packet) subject to delivering any offered traffic load that lies within the network's capacity region while maintaining bounded queues. For more details on stochastic optimization of wireless networks, we refer the reader to [10].

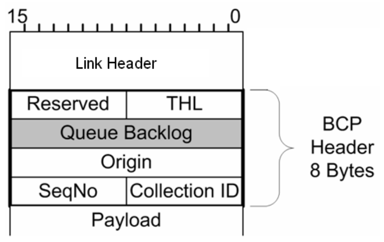

Fig. 2: The unicast data packet header in BCP. Note that all fields are identical to CTP [9], with the exception that CTP's ETX metric field has been used by BCP to broadcast the local node's packet backlog. The Time Has Lived (THL) field tracks hop count, while origin and seqno semi-uniquely define a packet. The Collection ID allows for multiple flow types to the sink. No special beacon/control packets are necessary for routing control in BCP.

Both the expected number of transmissions $ETX_{i,j}$ and rate $R_{i,j}$ are estimated in an online fashion by each node i for each of its neighbors based on local time stamps of its unicast data transmission attempts and corresponding received acknowledgements, using exponentially weighted moving averages (BCP uses a simple stop-and-wait ARQ with a maximum of 10 retransmission attempts on a link before weights are recomputed). For our EWMA estimates, we use the same $\alpha = 0.9$ setting as determined best by CTP. This setting gives good estimate of stability, at the cost of lesser responsiveness to sudden link changes. Each data packet includes a header field for the queue backlog (see figure 2). This information is obtained by all neighbors using a snoop interface. Thus BCP incurs no additional overhead in terms of separate broadcast control packets for either link estimation or for exchanging queue status. The parameter τ, which determines the time for which a packet that is not forwarded is withheld before the metric is recalculated, provides a tradeoff between throughput/delay performance and processor loading.

Unlike path-based routing protocols, in BCP, some packet looping is acceptable in response to temporary congestion/hot spot formation (though this is minimized as a natural consequence of reduced transmission counts). It is sufficient therefore to suppress only the duplicate packets that occur due to lost acknowledgements on a link. This is done by recording the three-tuple of $<$ source, sequence number, THL $>$ from the header of the last packet obtained from each incoming neighbor and using it to drop any duplicates with the exact same tuple.

IV. Testbed Results

A. Experimental Setup

We perform our evaluation experiments on the Tutornet [2] 40-node indoor wireless sensor network testbed consisting of IEEE 802.15.4-based Tmote Sky devices. A transmit power of -18 dBm was used for all experiments over 802.15.4 channel 26. Packet inter-arrival times were exponential, providing poisson traffic, and all tests were run for 35 minutes. In all tests, 39 motes were sourcing traffic. For brevity, we will state only the per source packet rate below, with the understanding that 1.0 packets per second indicates 39 sources are each active at this rate. The backpressure optimization parameter V was set to 2 as a result of early experimentation. We conservatively set $\tau = 100ms$.

B. Goodput

We first explore the delivery capabilities of CTP, BCP and Volcano. Figure 3 provides the goodput at the sink over various source rates. At source rates in excess of one packet per second we begin to see packet losses over CTP, indicating the need for source congestion control. The BCP performance, however, indicates no significant losses until source rates exceed 1.66 packets per second per source, a more than 50% improvement.

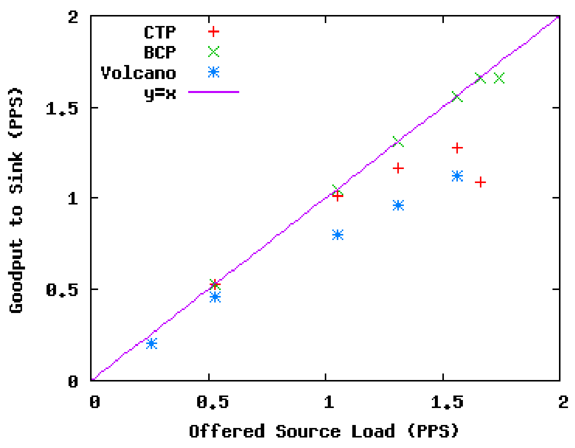

Fig. 3: Goodput over source rate for CTP, BCP and Volcano.

The cause of losses can be seen by observing maximum queue occupancy, shown in Figure 4 for 0.50 packets per second (top) and 1.50 packets per second (bottom). We note that motes near the sink in CTP are more prone to queue overflow, and that this causes aggressive tail drops at source rates in excess of 1.0 packets per second. Figure 4 also clearly depicts the natural queue gradient that results from backpressure routing: nodes near the sink have lower queue sizes, and queues grow the further a node is from the sink.

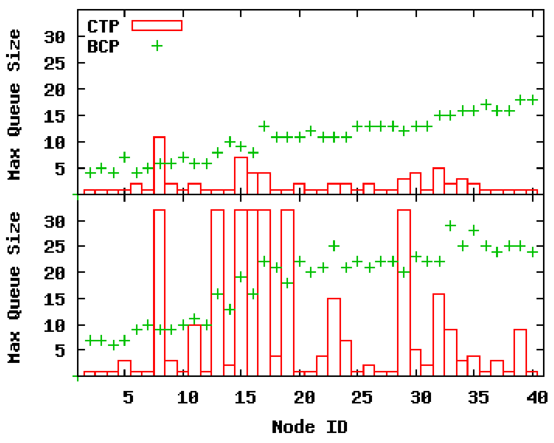

Fig. 4: Maximum queue sizes for the 0.50 packet per second tests (at top) and 1.50 packet per second tests (at bottom). We observe the hot spots spreading in CTP, while BCP exhibits the expected queue behavior that grows further from the sink.

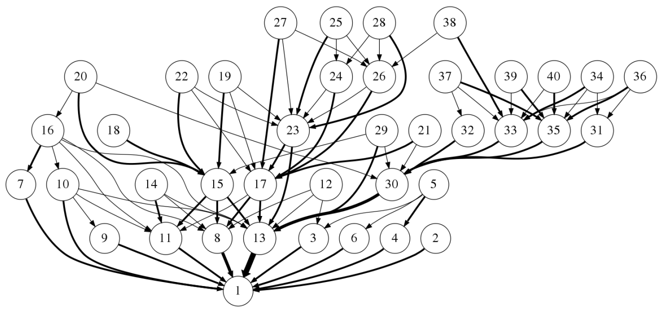

Fig. 5: The BCP routing result at 0.5 packets per second, link line weights indicate packet count transmitted in the 35 minute test.

This queue load balancing results in a spreading of traffic across the network for BCP. This is indicated by Figure 5 which summarizes the routing activity of BCP over a period of time. The tree structure of CTP on the other hand results in hot spots such as node 8 in the top plot of Figure 4. Though in experimentation we observed Volcano load balancing in a similar manner, it does not achieve similar goodput performance. We next set out to investigate the cause for this gap.

C. Delivery Efficiency

Figure 6 gives statistics for average hop count per packet for CTP, BCP and Volcano over source rates ranging from 0.25 packets per second through 1.5 packets per second. Both Volcano and BCP perform well with regard to hop count, particularly at low data rates. This can be explained with the simple observation that links are relatively reliable when the system is not heavily loaded; minimizing ETX is then approximately equivalent to minimizing hop count.

If Volcano is properly minimizing hop count, and the underlying backpressure optimization is designed to maximize throughput, why is the goodput to the sink suffering? Figure 7 plots statistics for average transmissions per packet under CTP, BCP and Volcano. Here we see the key problem with Volcano. In minimizing the hop count, often links are selected which

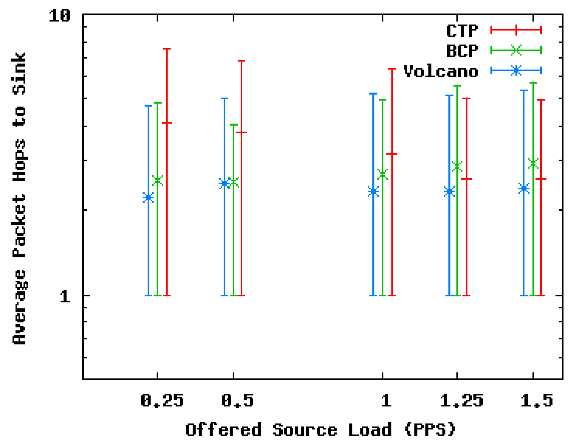

Fig. 6: The system average hop count per packet to the sink. The bars indicate the minimum and maximum source's average hop count per packet. Both BCP and Volcano perform competitively.

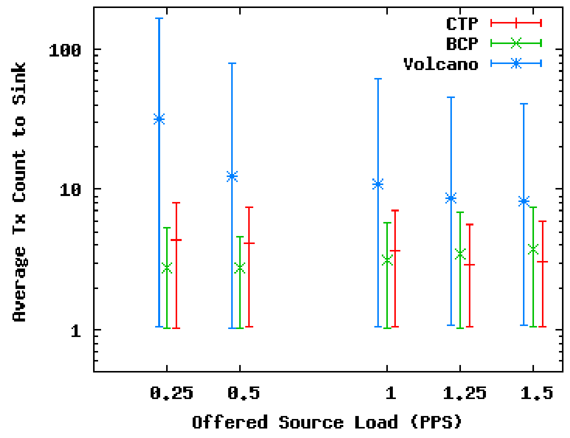

Fig. 7: The system average transmissions per packet to the sink. The bars indicate the minimum and maximum source's average transmissions per packet to the sink. BCP optimizes for system transmissions effectively, while Volcano selects links with poor quality.

have poor reliability, resulting in high transmission counts to the sink. Unfortunately, this problem is compounded by false acknowledgements, which occur in 802.15.4 networks acutely at high transmission rates due to the standard's use of low-complexity acknowledgements. These together cause the loss of approximately 20% of the packets across all source rates. The prior paper on Volcano routing [8], lacking a realistic link model, did not uncover these issues.

Also of interest in Figure 7 is the impressive performance of BCP with respect to total system transmissions. This metric is frequently selected by researchers in order to minimize energy consumption in wireless sensor network deployments, and we observe competitive performance versus CTP. Note that at source rates above 1.0 packets per second, queue tail drops in CTP disproportionately impact packets from the rear of the network, giving CTP artificially low averages for hop and transmission counts.

D. *Delay Performance*

Finally, we evaluate the improvement achieved by replacing BCP FIFO queues with LIFO ones. Figure 8 gives the range of per source average delay for per source rates ranging from 0.25 pps through 1.50 pps. At low data rates, we observe that BCP-FIFO delay performance scales inversely with source rate. This is because the data in the FIFO queues must be pushed through the gradient of queues in order to reach the sink, propelled by subsequent source admissions (recall this was the case in Figure 1). Replacing the FIFO queues with LIFO queues yields a delay reduction of more than 98% at light source rates (0.25 packets per second), and more than 75% reduced at heavy source rates (1.5 packets per second). These reductions make BCP competitive with CTP.

A traditional concern over LIFO queue usage is its impact on the ordering of packets at the final destination. Though not all sensor network applications are affected by packet re-ordering, there are currently some proposals for running TCP over IP within wireless sensor networks [1], the performance of which would suffer greatly from packet re-ordering if severe. We were therefore interested in comparing the re-ordering impacts of LIFO and FIFO queue usage in BCP. Figure 9 provides a typical 200 packet window of source sequence number versus sink arrival number for BCP FIFO and BCP LIFO under a high (1.5 packet per second) source rate. As would be expected, we do observe that the greatest re-ordering outliers exist under LIFO implementation. Somewhat surprisingly, however, we see that the majority of delivered packets experience lesser reordering under LIFO usage. The explanation is actually quite simple: under FIFO usage the delay variation from source to source has been seen to be extreme in figure 8. Subsequently, as BCP elects to load balance packets to different neighbors, significant re-ordering can result. Under LIFO queue usage, the delay to the sink is generally compressed (most packets traverse the hops rapidly) and so the dynamic routing actually has a lesser impact.

V. Related Work

The intellectual roots of dynamic backpressure routing for multi-hop wireless networks lie in the seminal work by Tassiulas and Ephremides [14]. They showed that using the product of queue differentials and link rates as link weights for a centralized maximum-weight-matching algorithm allows any traffic within the network's capacity region to be scheduled stably. Recent work by Neely [10] has provided a theoretical framework for backpressure-based stochastic optimization that we have used in this work to derive the link-specific thresholds for Volcano and BCP-ETX. Unlike most prior theoretical work focused on centralized TDMA-based backpressure scheduling, BCP works over an asynchronous CSMA MAC to provide a more practical distributed approach.

A closely related work to ours is volcano routing, a previous proposal for backpressure routing [8]. Volcano routing is designed and evaluated only in simulations assuming idealized loss-less links and global synchronization, and neither its goodput nor its delay performance are explored in [8]. As

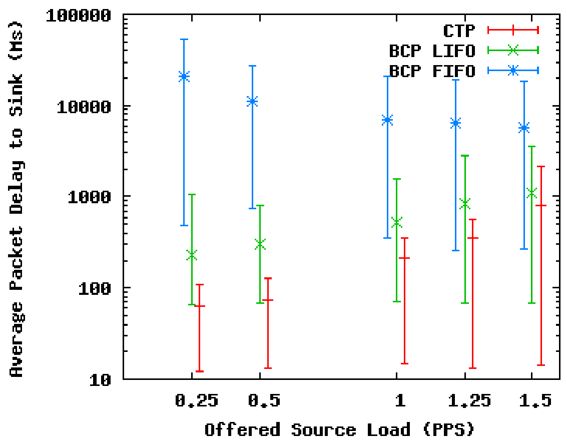

Fig. 8: The system average source to sink packet delay for CTP, BCP FIFO and BCP LIFO. The bars indicate the minimum and maximum source's average source to sink packet delay. The usage of LIFO queues in BCP results in a 75-98% reduction in average delay over FIFO.

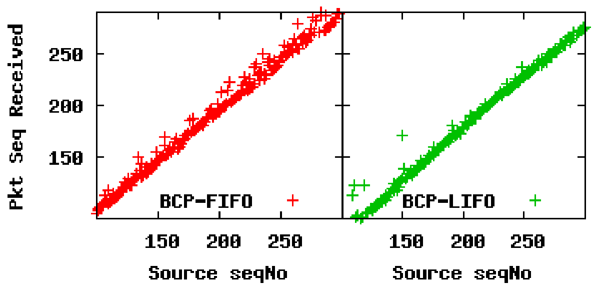

Fig. 9: Packet reception order at the sink for source 40, plotted for CTP, BCP FIFO and BCP LIFO. Interestingly, the bulk of packets received at the sink have less reordering under LIFO usage, with a few exceptional outliers which may be discarded.

we have shown, while it can provide low hop-counts, using the Volcano-like link weight shows poor performance in a real-world network in terms of transmission counts as well as goodput.

In recent years backpressure scheduling and optimization have been translated to practical protocols for wireless networks by a few other researchers, but only at the MAC and Transport layers. In wGPD [3] and DiffQ [15], the queue differentials are used to change contention behavior at the MAC layer. In Horizon [12], load balancing decisions over multiple disjoint routing paths (generated separately by a link state routing protocol) take into account queue state information to enhance TCP performance. Transport layer backpressure-based rate utility optimization is demonstrated in both wGPD [3] and in the work by Sridharan *et al.* [13]. None of these prior works have implemented dynamic backpressure *routing* at the network layer, which is the focus of BCP. Moreover, these prior works have not investigated the mitigation of delay in backpressure based protocols, a key contribution of this work.

While there have been previous theoretical/simulation-based proposals for use of multi-path routing in wireless sensor networks [7], nearly all implemented routing protocols for collection in wireless sensor networks have been based on minimum cost trees, including the state-of-the-art collection tree protocol (CTP) [9], which we have used as a baseline in our evaluation.

VI. Future Work

One of the most exciting aspects of our work with BCP is the number of extensions available for future research and development, both by our group and others. *We believe that BCP can be the basis of a comprehensive new high-performance cross-layer networking stack for wireless sensor networks*. In light of this potential, we have implemented BCP in the open source TinyOS environment.

Some immediate extensions that we plan to pursue pertain to providing automated parameter adaptation (for protocol parameters such as V and τ), and to more thoroughly evaluating the performance of BCP with respect to stochastic dynamics in link conditions, external interference, and the network topology. With a backpressure routing stack in place, it is very easy to implement transport layer congestion control on top that allows for the maximization of any concave source-rate utility function. We plan to do this in the future, similar to the backpressure-based transport layer optimizations implemented in ([3], [13]). We also plan to investigate if there are any further throughput gains to be obtained with MAC-layer prioritization based on the back-pressure weights (as has been explored in [15], [3]). Another desirable extension would be to integrate BCP over a suitable multi-frequency MAC.

In future work, it is also of interest to explore how receiver diversity mechanisms such as used in the ExOR protocol [5] may be incorporated within the BCP framework to further improve network throughput, delay and energy performance. The recently proposed DIVBAR mechanism [11], which has been analyzed theoretically, suggests that this is feasible. Also exciting is the prospect of implementing network coding techniques with BCP, taking advantage of the routing path diversity provided by backpressure. This may make it possible to give some guarantee of source to sink loss rates, allowing high probability packet reconstruction at the sink while removing link layer acknowledgement overheads in the 802.15.4 MAC.

References

[1] IETF 6lowpan Working Group. *http://www.ietf.org/dyn/wg/charter/6lowpan-charter.html*.

[2] Tutornet Testbed. *http://enl.usc.edu/index.html*.

[3] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Sanjee, and A. Stolyar. "Joint Scheduling and Congestion Control in Mobile Ad-hoc Networks". *IEEE Infocom*, 2008.

[4] M. Bathula, M. Ramezanali, I. Pradhan, N. Patel, J. Gotschall, and N. Sridhar. "A Sensor Network System for Measuring Traffic in Short-Term Construction Work Zones". *DCOSS*, 2009.

[5] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. *ACM SIGCOMM Computer Communication Review*, 35(4):133–144, 2005.

[6] D.S.J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Mobicom*, 2003.

[7] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25, 2001.

[8] Y. Ganjali and N. McKeown. Routing in a highly dynamic topology. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pages 164–175, 2005.

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. “Collection Tree Protocol”. *To appear, ACM Sensys*, 2009.

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

[11] M.J. Neely and R. Urgaonkar. Optimal backpressure routing for wireless networks with multi-receiver diversity. *Ad Hoc Networks*, 7(5):862–881, 2009.

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

[13] A. Sridharan, S. Moeller, and B. Krishnamachari. Making distributed rate control using Lyapunov drifts a reality in wireless sensor networks. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*, pages 452–461, 2008.

[14] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.

[15] A. Warrier, S. Ha, P. Wason, I. Rhee, and J.H. Kim. Diffq: Differential backlog congestion control for wireless multi-hop networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*, pages 585–587, 2008.