# Making Distributed Rate Control using Lyapunov Drifts a Reality in 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

***Abstract*—We take a top-down approach of formulating the rate control problem, over a collection tree, in a wireless sensor network as a generic convex optimization problem and propose a distributed back pressure algorithm using Lyapunov drift based optimization techniques. Primarily, we show that existing theoretical results in the field of stochastic network optimization can be directly applied to a CSMA based wireless sensor network using our novel receiver capacity model. We back this claim by implementing our algorithm on the Tmote sky class devices. Our experimental evaluation on a 5 node testbed shows that the empirically observed rate allocation on a real sensor network testbed that uses our back pressure algorithm is close to the analytically predicted values, justifying our claims.**

## I. Introduction

In recent years, the literature on wireless networks has been enriched by several theoretical results that have developed new mathematical frameworks for design of optimal cross-layer protocols [2]. A particularly appealing stochastic network optimization approach that yields simple distributed algorithms for dynamic scenarios is based on the use of Lyapunov drifts [4].

The Lyapunov drift based techniques described in [4] provide for distributed rate control based purely on local observations of neighborhood queues. They guarantee the stability of the system and also provide mechanisms to achieve optimization with respect to given utility functions. While attractive on theoretical grounds, to our knowledge these techniques have yet to be implemented in real wireless networks [1]. One reason it has not been easy to translate the Lyapunov drift methodology from theory to practice is that it has been primarily developed under the assumption of a time-slotted system, implying a TDMA MAC. TDMA-based wireless networks are generally harder to implement due to the challenges associated with time-synchronization, particularly across multiple-hops.

Our primary contribution in this work is to show that rate control algorithms based on the Lyapunov drift framework can be built over asynchronous CSMA-based MAC protocols. Specifically, for a wireless sensor network, we model the available bandwidth using the concept of receiver capacity [13]. This model introduces virtual queues in the Lyapunov drift framework that capture the interference constraints existing in the network.

Our second contribution in this work is the experimental implementation of this distributed queue-based rate control algorithm on a real low-power wireless platform (the Tmote Sky from Moteiv) over a CSMA MAC (the CC2420 communication stack implemented in TinyOS 2.x). We provide details on our implementation and present experimental results which validate our analysis, bringing Lyapunov-based rate control algorithms for wireless networks a step closer to reality.

## II. Related Work

The problem of rate control in wireless sensor networks has seen a rich set of proposals from a systems perspective([3], [5], [9], [10], [11], [12], [18], [19]). Most of these protocols are designed purely from the perspective of congestion control and hence focus primarily on network stability. Many of these protocols implicitly or explicitly aim at providing some notion of fairness while achieving congestion control. However, to our knowledge, there is no prior work on practical rate control mechanisms for wireless sensor networks that provide the ability to optimize any generic convex utility function.

Our contribution to the body of work on rate control in wireless sensor networks is to advocate a top-down approach for designing rate control algorithms. In this work, we first formulate the problem of rate control over a collection tree in wireless sensor networks, as a generic convex utility optimization problem. Using the Lyapunov drift framework, we then present a distributed algorithm and a proof of concept systems implementation that solves the convex optimization problem.

Tassiulas *et al.* in [16] and [17] first introduced the design of backpressure algorithms based on Lyapunov drift techniques to achieve network stability. The work by Neely *et al.* ([7], [8]), and Stolyar *et al.* ([15]) significantly builds on the original work by Tassiulas*et al.* by showing that Lyapunov drift based back-pressure algorithms can be designed not only to achieve network stability but also to optimize system wide utility, as long as the system wide utility function is convex. The framework presented in [7] and [8] allows for the transformation of the constrained optimization problem to a queueing theoretic problem by associating a virtual queue with each constraint in the original optimization problem. Optimizing the drifts of

This work is supported in part by NSF grants numbered 0347621, 0627028, 0430061 and 0325875

[1]Although we have anecdotal evidence that an implementation of similar theoretically-derived queue back-pressure algorithms is being attempted in an ongoing DARPA-funded project, we are not aware of any prior published implementations of such techniques.

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.

DOI 10.4108/ICST.WIOPT2008.3205

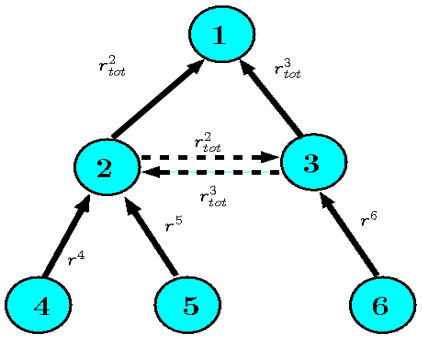



Fig. 1. An illustrative example of the receiver capacity model

the physical and and virtual queues then presents a tradeoff between the stability of the system and utility optimization.

There exists a rich literature on using Lyapunov drift based approaches for solving problems of optimal power control, rate control and energy optimization in wired and wireless networks which has been covered in [4]. Despite this theoretically well grounded literature, specifically in the domain of wireless ad-hoc networks, we are not aware of examples of real systems employing the strategies designed using the Lyapunov drift based approach. This is primarily due to the underlying assumption of a slotted system, implying a TDMA MAC. Our main contribution in this domain is to present a technique of implementing a Lyapunov drift based backpressure algorithm for a CSMA based system in a wireless sensor network setting. We are able to achieve this objective due to the linear constraints presented by our receiver capacity model, which allows these constraints to be easily modeled as a set of virtual queues. Thus, the problem of designing backpressure algorithms using Lyapunov drifts in a CSMA setting is tractable.

## III. Problem Formulation

In wireless sensor networks, the dominant topology is a collection tree where multiple sources are forwarding data to a single sink. We consider the following optimization problem over a collection tree.

$$\max : \sum_{\forall\ i} g_i(r_i)$$
$$s.t.\ r_i\ \in \Lambda$$

Where $r_i$ is the time average source rate for each source $i$, $g_i(r_i)$ is assumed to be convex and $\Lambda$ is the capacity region for the collection tree. To solve the above optimization problem we need to the know the capacity region $\Lambda$ which constrains the optimization problem.

We use the receiver capacity model presented in [13] in order to define the capacity region for a wireless sensor network collection tree leveraging a CSMA based MAC. The core idea is to associate a constant bandwidth capacity with each receiver in the network. This capacity must be shared by all transmitters within interference range of that receiver. In particular, for any node $i$, the rates allocated to a) all nodes sending data to that node $i$, b) all transmitting nodes within interference range of $i$, and c) the transmissions made by node $i$, must not exceed that node's receiver capacity. This model corresponds to a linear approximation of the capacity region for each receiver. That is, any linear combination of neighborhood transmission rates is feasible so long as the net overheard rate does not exceed the receiver bandwidth. Intuitively, one would expect such a linear rate region approximation to be reasonable for CSMA (operating on small packet sizes $\sim$ 40 bytes) precisely because it minimizes collisions through carrier sense (yielding similar sum-rates for different levels of contention between a set of users). We have previously validated the appropriateness of this approximation through experiments with real wireless devices [14].

We present an illustrative example using Figure 1 to highlight the applicability of the receiver capacity model in defining the constraints of the above optimization problem.

Figure 1 shows a 6 node topology. The solid lines represent the collection. The dashed lines quantify the interference existing in the network. For example, when node 2 sends data to node 1 at some rate, node 2 not only consumes the corresponding amount of capacity at node 1 but also at node 3. This is indicated by setting equal rates on the links $2 \rightarrow 1$ and $2 \rightarrow 3$.

For Figure 1, based on the constraints generated by the receiver capacity model [13], the optimization problem can be rewritten as:

$$\mathbf{P1} : \max \sum_{\forall\ i} g_i(r_i)\ i \in \{2,3,4,5,6\} \quad (1)$$
$$r^2 + r^3 + r^6 + 2r^4 + 2r^5 \leq B^2 \quad (2)$$
$$r^2 + 2r^3 + r^4 + r^5 + 2r^6 \leq B^3 \quad (3)$$
$$r^2 + 2r^4 + r^5 \leq B^4 \quad (4)$$
$$r^2 + r^4 + 2r^5 + r^5 \leq B^5 \quad (5)$$
$$r^3 + 2r^6 \leq B^6 \quad (6)$$

In Figure 1, $r^2_{tot}$ and $r^3_{tot}$ are given by:

$$r^2_{tot} = r^2 + r^4 + r^5$$
$$r^3_{tot} = r^3 + r^6$$

## IV. Lyapunov Optimization Formulation

Our objective is to find a distributed algorithm to solve the optimization problem presented in Section III.

By modeling optimization problem constraints as virtual queues, prior work in the area of stochastic network optimization ( [6], [8]) presents techniques minimizing the drift of a linear combination of the physical and virtual queues of the whole system, thereby ensuring forwarding queue stability while obeying constraints. The objective function may be incorporated as a penalty or reward function included in the drift bound, providing a final solution which trades system queue size and latency for utility optimality. The modularity of the algorithms resulting from this approach is one of its primary attractions.

Prior work such as ( [6], [8]) assumed a detailed knowledge of the physical layer channel capacity. This was then used by a possibly centralized channel optimizer in order to ascertain

optimal transmission rates per the Lyapunov drift minimization algorithm. The implicit assumption in the allocation is that the underlying MAC is TDMA. Though there is nothing in the analysis that limits the methodologies to a TDMA based approach. The Achilles' heel of this approach seems to be that the optimization agent must have knowledge of the physical layer capacity region. The novelty of the solution presented here is that, using existing approaches, we show that the optimization problem can be applied to a CSMA based network as well. This is achieved by the additional constraints to the optimization problem which use the receiver capacity model, and by relaxation of the exact channel capacity assumption in optimization over $X_i(t)$.

In this section, we present a Lyapunov drift based solution to the problem $\mathbf{P1}$ presented in Section III.

### *A. Lyapunov Optimization with Receiver Capacity Virtual Queues*

Definitions of variables used in this Lyapunov formulation are given in Table I. For our Lyapunov drift formulation we assume that the system operates on slot boundaries of duration $T$ seconds. For analytical tractability we assume global synchronization between the nodes. We will relax this assumption when we describe our protocol implementation. As mentioned in summary earlier, the strength of our technique lies in decoupling the physical channel capacity region from the transmission rate decisions ($X_i(t)$s). We can therefore abstract the channel capacity of our Lyapunov optimization as follows: we assume that all nodes can transmit simultaneously without interference, and support only two transmission values. In a given slot $t$, each $X_i(t)$ is set to one of $\{0, B_{max}\}$, with $B_{max}$ a constant parameter in the deployment, likely set to a value marginally greater than the maximum receiver bandwidth of any node in the network. This way, nodes toggle between on and off modes of operation independently, with no concern for neighboring node's activities. We rely on the receiver capacity model constraints to enforce stability over the CSMA channel.

Using the Lyapunov drift approach, we first convert each of the constraints in the problem $\mathbf{P1}$ to a virtual queue. Since a constraint is associated with each node $i$ (since every node in the network is a receiver), we associate a virtual queue $Z_i$ with each node.

The queuing dynamics for each of the virtual queues $Z_i(t)$ is given as follows:

$$Z_i(t+1) = \max[Z_i(t) - B_i, 0] + \sum_{j \in D_i} \hat{X}_j(t) \qquad (7)$$

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

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

$$U_i(t+1) = \max[U_i(t) - X_i(t), 0] + \sum_{j \in C_i} \hat{X}_j(t) + R_i(t) \quad (8)$$

That is, each node $i$ first attempts to transmit $X_i(t)$ units of data to its parent, then receives $\hat{X}_j(t)$ units of data from each child node $j$. Note that we differentiate here attempted transmissions ($X_i(t)$) and true transmissions ($\hat{X}_i(t)$). The difference being that while it may be most optimal to transmit a complete $X_i(t)$ units of data in this timeslot, the queue may not contain sufficient data to operate optimally, so $\hat{X}_i(t) \leq X_i(t)$.

Combining the objective function $\sum_{\forall\ i} g_i(r_i)$ with the queueing dynamics presented in equations (7) and (8), we can perform a Lyapunov drift optimization that will result in an algorithm that has two components: a control decision and an admission decision. Each decision will be performed by every node in the network at each time step. A node performs a control decision to determine whether it is optimal to forward packets up the collection tree. The admission decision is performed in order to determine if a local application layer packet should be admitted to the forwarding queue. For ease of exposition, we refer the reader to Appendix A for details of the Lyapunov drift analysis. We will now proceed to explain the control and admission decisions in further detail.

*1) Control Decision :* The control decision for a node $i$ with a parent $k$ is the following:

$$\left[U_i(t) - U_k(t) - \hat{Z}_i(t)\right] \geq 0 \qquad (9)$$

If condition (9) is true, maximize $X_i(t)$ by setting it to $B_{max}$. As mentioned earlier, the detailed derivation of the condition presented in equation (9) are presented in Appendix A. A node transmits data to the parent if and only if the differential backlog between the node and its parent exceeds the sum of virtual queues within the local node's neighborhood.

*2) Admission Decision:* The local admission decision for a node $i$ is based on selecting $R_i(t)$ so as to maximize the following:

$$\left[\frac{V_{opt}}{2} \cdot g(R_i(t)) - U_i(t) \cdot R_i(t)\right] \qquad (10)$$

The derivation of this admission decision has also been presented in Appendix A. Node $i$ then selects a volume of local admissions in timeslot $t$ equal to $R_i(t)$ such that expression (10) is maximized.

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

An intuition for this behavior of $V_{opt}$ can be obtained by looking at the feasible solutions of the optimization problem $\mathbf{P1}$. In the optimal solution of $\mathbf{P1}$, all the constraints in $\mathbf{P1}$ need to be tight. This implies that the system needs to be at the boundary of the capacity region, which further implies that system will be unstable (queue sizes will be unbounded). If we want to keep the system stable, we need to keep the

| Symbol | Description |
|---|---|
| $U_i(t)$ | The queue backlog for node i at time slot t |
| $Z_i(t)$ | Virtual queue backlog for node i's collision domain at time slot t |
| $X_i(t)$ | The attempted transmissions up the tree by node i in time slot t |
| $\hat{X}_i(t)$ | The actual transmission rate up the tree by node i in time slot t |
| $R_i(t)$ | The admitted exogenous arrivals to node i in time slot t |
| $B_i$ | The receiver capacity of node i |
| $D_i$ | The collision domain of node i, includes neighbors and node i |
| $C_i$ | The set of one-hop children for which i is the forwarding agent |
| $\hat{Z}_i(t)$ | The sum of all virtual queues within i's collision domain in time slot t $\hat{Z}_i(t) = \sum_{\forall j \; st \; i \in D_j} Z_j(t)$ |
| $V_{mult}$ | The virtual queue multiplier, scales virtual queues for comparison with forwarding queue backlogs |

TABLE I
VARIABLES USED IN THE LYAPUNOV FORMULATION.

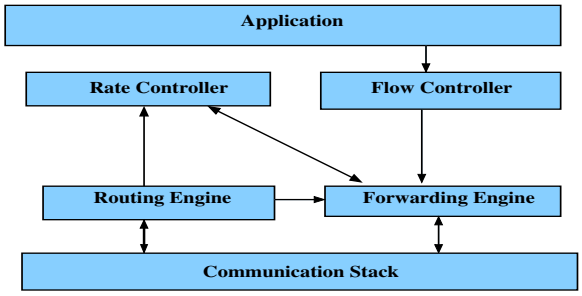



Fig. 2. Software architecture for the distributed rate control using Lyapunov drifts on TinyOS 2.0.2

constraints loose. This requires that the system to achieve a suboptimal solution with respect to the objective function, but ensures stability. Thus, $V_{opt}$ tunes how closely the algorithm operates to the boundary of the capacity region.

## V. SYSTEMS IMPLEMENTATION

### A. Hardware / Software Implementation Details

Our target platform was the Tmote sky class of devices. As mentioned earlier, our receiver capacity model has been validated empirically for these devices [14]. Tiny OS is the open source operating system that runs on the Tmote sky and hence our software architecture was designed specifically to work with the TinyOS infrastructure.

Figure 2 presents the software architecture of our implementation. Because the objective was to perform distributed rate control over a static tree, we use the routing engine provided by the collection tree protocol in TinyOS-2.0.2 [1]. The routing engine helps build a tree rooted at a specific node, acting as the sink. The rate controller block implements the control decision given by condition (9). Though our analysis, presented in Section IV, assumes that all events occur on slotted time boundaries, this is not the case for a real CSMA based MAC. Hence the rate controller implements a timer that fires every $T$ seconds. The control decision are thus made at the end of $T$ seconds. It's essential to note that these timers are local to a node, making the decisions asynchronous.

The rate controller block also estimates the node's current transmission rate by maintaining an exponential weighted moving average of the number of packets transmitted by the forwarding engine, on a per slot basis. In addition to rate estimation, the rate controller block updates the local virtual queue using equation ( 7). This requires knowledge of the local node's receiver capacity, its current transmission rate and the transmission rate of all its neighbors who are active in its broadcast domain. The rate controller establishes its receiver capacity by setting it to the saturation throughput corresponding to the number of neighbors within its broadcast domain [14]. In this paper, we have hard-coded the receiver capacities to 70 packets per second, a safe lower bound to the optimal capacity. Techniques could be implemented that would improve system performance by estimating the number of active neighborhood transmitters in each time slot. We feel that for our basic proof of concept, such techniques lie outside the scope of this paper.

The flow controller block implements the admission decisions. Though the Lyapunov analysis of Section IV makes no assumptions on the form of the utility function other than convexity, in order to simplify admission rate computation we limited our laboratory testing to linear utility functions. By assuming linear $G(R_i(t))$ we can simplify our admission decisions. Let $g(R_i(t)) = u_i \cdot R_i(t)$. Note that this reduces our admission decision components of the Lyapunov drift bound to the following:

$$-R_i(t) \cdot \left[ \frac{V_{opt}}{2} \cdot u_i - U_i \right] \tag{11}$$

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

The forwarding engine maintains the forwarding FIFO queue. This queue receives packets both from local admissions and from children in the tree structure. The communication stack consists of the default TinyOS CSMA stack for the CC2420 radios which is the 802.15.4 radio present on the Tmote sky platforms. Note that in order to carry out the control decision based on condition ( 9), knowledge is needed of the number of transmissions ($X_i(t)$) and virtual queue sizes ($Z_i(t)$) of all neighbors during the prior time slot. Therefore, once every time slot, each node attaches their transmission rate and virtual queue size to the next transmitted MAC header and broadcasts the data packet instead of unicasting it. This

allows all neighboring nodes to overhear this information and retrieve it from the headers. As mentioned in Section IV-A, control decisions are carried out independently, without inter-communication between nodes. Our implementation therefore allocates transmission tokens to the forwarding Engine in an all-or-none manner. Each node based on its control decision will either allocate $B_{max}$ or zeros forwarding tokens for the next time slot. This is where the virtual queues become critical, as they ensure that our time average receiver bandwidth constraints are not violated, even while individual nodes attempt to transmit at maximum rate during their active time slots.

### B. Virtual Queue Multipliers

Note that the discrete time queueing dynamics we have supplied for $Z_i(t+1)$ in equation (7) first service the queue, then incorporate arrivals. This implies that even in an extremely stable system, where the queue empties every timeslot prior to arrivals, the virtual queue will hold a continuous backlog equal to the transmissions overheard within the node i's neighborhood. Even in relatively small neighborhoods, this could be large enough to overwhelm our real queues in the control decisions dictated by condition (9). The result would be on-off behavior, where any transmissions in timeslot $t$ would result in large virtual queues and silencing of transmissions in timeslot $t+1$, only to have the virtual queues then emptied and transmissions allowed in timeslot $t+1$. This oscillatory behavior is clearly non-optimal, and is an artifact of placing too much relative weight on the virtual queues, representing time average receiver capacity violation.

There are two solutions to this problem. One is to increase the value of $V_{opt}$. This directly leads to larger queue backlogs, and therefore mitigates the effects of (relatively) smaller virtual queues, allowing for greater system throughput. In wireless sensor networks, however, queue capacities are quite limited. Storing more than fifty packets can become a challenge. We therefore introduce a second optimization parameter, the virtual queue multiplier $V_{mult}$. The control decision of condition (9) then becomes:

$$X_i(t) \cdot \Big[U_i(t) - U_k(t) - V_{mult} \cdot \hat{Z}_i(t)\Big] \qquad (12)$$

### C. An Illustrative Example

We now provide a simple illustrative example to give some intuition on the admission and forwarding control decisions. Consider a five node linear forwarding topology as in topology 1 of Figure 4. All nodes are considered to be traffic sources (omitting the sink, node 1). All nodes are assumed to be within range of one another, implying that interference links exist between all nodes. Assume a linear $g(R_i(t)) = u_i \cdot R_i(t)$. Let $[u_5, u_4, u_3, u_2, u_1] = [3, 6, 2.5, 1, 0]$, $V_{opt} = 20$, $V_{mult} = 1$, and $B_{max} = 1$. For this example, we assume that we have infinite sink-destined data at each of the sources.

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

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

Recall from equation (11) that if the queue backlog is greater than $\frac{V_{opt}}{2} \cdot u_i$, we reject the application layer admission request. This implies that local admissions will halt above predetermined queue backlog thresholds, equal to $\lfloor \frac{V_{opt}}{2} \cdot u_i \rfloor$. For the current topology, queue thresholds for $[U_5, U_4, U_3, U_2]$ are therefore $[30, 60, 25, 10]$.

Figure 3(a) depicts the early startup phase. In this phase, the virtual queues are small because the nodes have only recently been in violation of the time average receiver capacity constraint. Because the virtual queues are small, condition ( 9) dictates that in order for nodes to transmit, the forwarding queues need not exceed the local admission thresholds. As the admission threshold is not being exceeded, we will see traffic from sources at nodes 3 and 2 entering the system. This is a suboptimal behavior.

Figure 3(b) depicts the system as it moves towards equilibrium. As transmission rates continue to violate receiver capacity, the virtual queues grow. This backpressure causes the forwarding queues to sequentially grow above their local admission thresholds. In this figure, for example, the value of $\hat{Z}_2(t)$ has grown to 12. A forwarding queue backlog of at least 12 packets is therefore required in order to transmit to the sink. This exceeds node 2's admission threshold of 10 packets. Therefore all future local admission request will be rejected by node 2.

Finally, Figure 3(c) depicts the system at equilibrium. Here, a special condition holds. The sum of $\hat{Z}_i(t)$ between node 4 and the sink is exactly equal to node 4's admission threshold of 60 packets. No other node in the network is allowed to admit local traffic, as the virtual queues have grown too large for any non-optimal source.

## VI. Experimental Evaluation

### A. Experimental Setup

In order to verify proper operation of our proposed algorithm, we performed experiments using the Tmote sky class devices running TinyOS-2.0.2. We considered three test scenarios. Two five node topologies were selected, as highlighted in Figure 4. Table II indicates the utility parameters

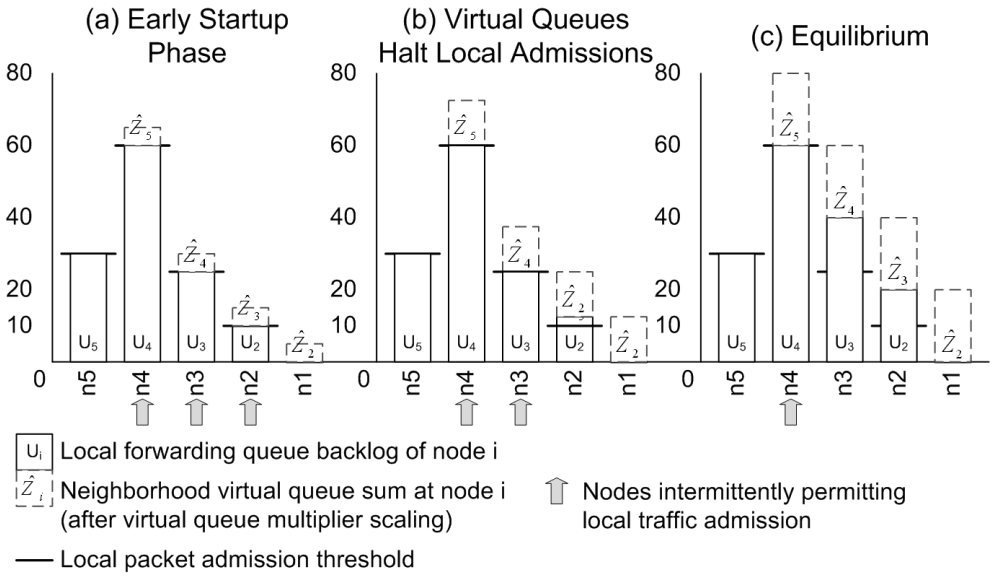



Fig. 3. An illustrative example of algorithm convergence to optimal admission and transmission policies on a linear, fully connected network with linear utility functions $g(R_i(t))$. From startup, the system progresses from (a) to (b), then on to (c) in equilibrium.

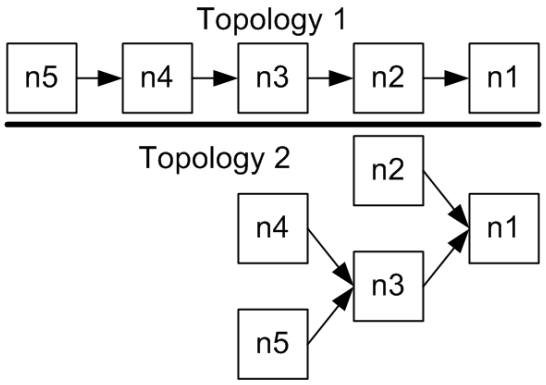



Fig. 4. Two test topologies utilized in experimentation.

| Scenario | $u_5$ | $u_4$ | $u_3$ | $u_2$ | Topology | Rcv Cap |
|---|---|---|---|---|---|---|
| 0 | 2 | 5 | 1 | 1 | 1 | 70 |
| 1 | 2 | 4 | 3 | 1 | 1 | 70 |
| 2 | 5 | 1 | 1 | 1 | 2 | 70 |

TABLE II
UTILITIES AND TOPOLOGIES FOR EACH OF THE THREE TEST SCENARIOS.

| Parameter | Value |
|---|---|
| Beacon Interval | 300 milliseconds |
| Token Allocation | 25 per Active Timeslot |
| $V_{opt}$ | 20 |
| Virtual Queue Multiplier | 0.01 |
| Per-node Receiver Capacity | 70 |

TABLE III
COMMON PARAMETERS FOR ALL TEST SCENARIOS.

and topology for each of the four test scenarios. Table III documents the parameters common to all scenarios.

These configurations were loaded onto five Tmote sky devices. In experimentation, each scenario was allowed to run for 25 minutes. Each node recorded the forwarding and virtual queue backlogs, while the root node recorded goodputs received from nodes in the system. In all tests, the networks were fully connected. We would like to extend the test environment to a larger system in future work.

Using the receiver capacity model, as shown in Section III, based on the utilizes and topologies we can pre-compute the optimal rate allocation for each of the scenarios. Optimal per-node transmission rates for all scenarios are given in Table IV.

### B. Experimental Results

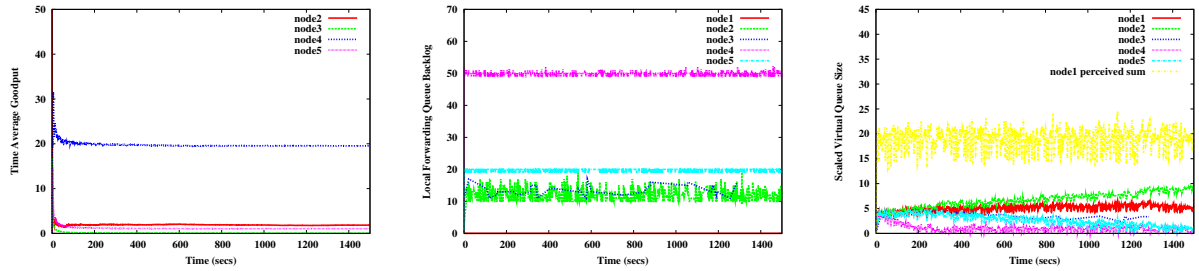
For each of the scenarios, the results of the four 25-minute experiments are plotted in Figures 5, 6 and 7. Time average goodputs have been recorded in Table V.

*1) Optimality Gap:* Comparison with the optimal solutions recorded in Table IV indicates that there is a consistent gap between achieved experimental results and LP optimal transmission rates.

Two possible explanations for this gap are as follows. First, there is some overhead required in our system implementation. As the root node never generates or forwards traffic, there would be no opportunity to broadcast virtual queue backlogs. We therefore require a broadcast packet be sent by the root node every second. This packet is then accounted for by the system and cuts into the receiver capacities of all nodes within

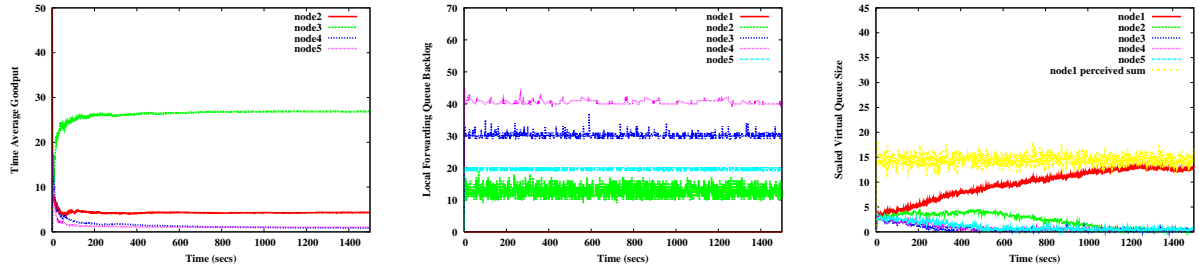
| Scenario | node5 PPS | node4 PPS | node3 PPS | node2 PPS |
|---|---|---|---|---|
| 0 | 0 | 23.3 | 0 | 0 |
| 1 | 0 | 0 | 35 | 0 |
| 2 | 35 | 0 | 0 | 0 |

TABLE IV
OPTIMAL PACKET RATE ALLOCATIONS SUBJECT TO RECEIVER BANDWIDTH CONSTRAINTS FOR SCENARIOS 0 THROUGH 2.



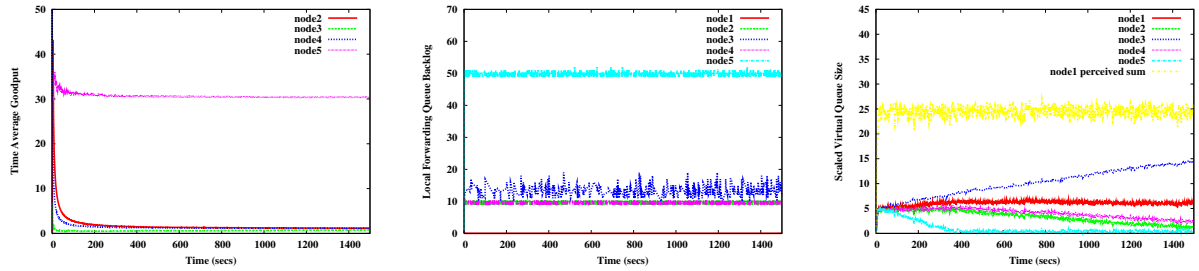

(a) Average goodput rates received at the sink. (b) Local forwarding queue sizes. (c) Local virtual queue backlogs.

Fig. 5. Plots of results for scenario 0.




(a) Average goodput rates received at the sink. (b) Local forwarding queue sizes. (c) Local virtual queue backlogs.

Fig. 6. Plots of results for scenario 1.




(a) Average goodput rates received at the sink. (b) Local forwarding queue sizes. (c) Local virtual queue backlogs.

Fig. 7. Plots of results for scenario 2.

the roots neighborhood. In our test environment, all nodes overhear the root, so all nodes incur costs within their virtual queues as a result of this mandatory broadcast. Additionally, consider that leaf nodes often are not optimal, and that nodes sitting behind the optimal traffic generator are very unlikely to ever forward or generate packets once equilibrium is reached. The effect of this, however, is that their broadcast updates containting real and virtual queue backlogs will halt, resulting in stale system state information. We therefore also require that all nodes not admitting or forwarding any traffic must still generate a minimum of one packet per second. The optimality gap is therefore variable and depends upon the number of inactive nodes.

Second, there is a true optimality gap between LP optimal and Lyapunov formulations and it is a function of $V_{opt}$ and $V_{mult}$ [4]. The derivation of this bound is part of our future work. This suboptimal behavior is clearly visible in scenario 1. Note that in Figure 6(a) node 2 receives 4.28 packets per second, though node 2 is not in the optimal solution. The optimal source node for this scenario is node 3 with $u_3 = 3$. Based on the control decision in equation(11), with a $V_{opt} = 20$, node 3 therefore admits packets up to a forwarding queue size of 30 packets. At equilibrium, as node 3 is two hops from the sink, this leads to a scaled neighborhood virtual queue backlog for node 3 equals 15, which is evident in Figure 6(c). Since the forwarding queue backlog at node 2 is no more then 15 and given our rather large per timeslot token allocations relative to these queue sizes, occasionally the queue of node 2 empties below 10. This is the admission threshold for node 2 and hence when forwarding queue size goes below this threshold, source traffic from node 2 is admitted. This leads to the suboptimal behavior observed in Figure 6(a). To make the rate allocation closer to optimal, we would need to enlarge our value of $V_{opt}$ such that the differential in forwarding queues between node 2 and node 3 would be sufficiently large. Increasing $V_{opt}$ was not an option in our setup, as the queues had already reached maximum sustainable values on our Tmote Sky device ( 50 packets).

*2) Virtual Queue Behavior:* From a theoretical standpoint, in a fully connected topology, all local virtual queues should have identical backlogs. However, in our experiments, note that all local virtual queues exhibit slow drifts away from their

| Scenario | node5 PPS | node4 PPS | node3 PPS | node2 PPS |
|---|---|---|---|---|
| 0 | 1.1 | 19.3 | 0.0 | 1.8 |
| 1 | 1.1 | 0.8 | 26.5 | 4.3 |
| 2 | 30.0 | 1.1 | 0.7 | 1.2 |

TABLE V
ACHIEVED PACKET RATE ALLOCATIONS FOR EXPERIMENTAL SCENARIOS 0 THROUGH 2.

theoretical values. The drift results from inaccurate updates of local virtual queues using stale neighbor transmission rate information. The stale neighbor transmission rate information occurs due to the asynchronous nature of the system. Despite the existence of this drift, the system performance is not affected because the performance does not depend on the individual virtual queue sizes but the sum of the virtual queues in the neighborhood. As the graphs reflect the sum of the virtual queues remains constant, thus no adverse effect on rate allocation is observed. Verifying proper behavior under virtual queue drift for systems with dynamic traffic flows is a topic for future investigation.

## VII. CONCLUSION AND FUTURE WORK

The primary goal of this work was to present a proof of concept which uses Lyapunov drift based techniques in a wireless sensor network setting. We have formulated the collection tree rate control problem as a constrained convex optimization using the receiver capacity model [13] for a CSMA based wireless sensor network . As a solution, we propose a back pressure algorithm designed using a Lyapunov drift based optimization framework. The back pressure algorithm provides near optimal utility while maintaining network stability. We have shown the efficacy of our proposed back pressure algorithm by implementing it on TinyOS-2.0.2 operating system for the Tmote sky class of devices. In our experimental evaluation over a 5 node testbed, we observe that the empirical behavior matches our analysis. This validates our claims.

In this work we have not characterized the effects of $V_{opt}$ on queue sizes and optimality. We plan on presenting theoretical bounds on the time average queue back logs, and bounds on achievable system utility. These bounds are dependent on the parameter $V_{opt}$. Theoretically, larger values of $V_{opt}$ result in utility that approaches optimality. This comes at the cost of larger queue sizes leading to larger end-to-end packet delays. It would therefore be worthwhile investigating the trade off, in terms of queue sizes and optimality, with smaller values of $V_{opt}$.

Extending this work further, an interesting future direction would be to investigate dynamic routing. In a CSMA setting, this work proves the applicability of existing stochastic optimization results that use Lyapunov drift based techniques. The literature in stochastic network optimization is rich with examples where, using the Lyapunov drift framework, back pressure algorithms can be designed that present a node in the network with not only control and admission decisions, but also routing decisions [8]. The primary advantage of such an algorithm is that it might result in higher throughput and would adapt to network dynamics. To our knowledge, there are no existing cross layer rate control protocols, involving dynamic routing, that perform utility optimization over a wireless sensor network.

## VIII. ACKNOWLEDGEMENT

We would like to acknowledge Prof. Michael J. Neely, who took the time and effort to introduce the authors to the technique of using Lyapunov drifts for the purposes of stochastic network optimization.

## REFERENCES

[1] *http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html*.
[2] M. Chiang, S.H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE, vol. 95, no. 1, pp. 255-312*, January 2007.
[3] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. *ACM Sensys*, 2004.
[4] 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.
[5] B. Hull, K. Jamieson, and H. Balakrishnan. Techniques for mitigating congestion in sensor networks. *ACM Sensys*, 2004.
[6] M.J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, 2003.
[7] M.J. Neely. Energy optimal control for time varying wireless networks. *IEEE Transactions on Information Theory*, 52(7), 2006.
[8] M.J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proceedings of IEEE INFOCOM*, 2005.
[9] J. Paek and R. Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. *Sensys*, 2007.
[10] 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.
[11] 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.
[12] A. Sridhara and B. Krishnamachari. Explicit and precise rate control in wireless sensor networks. *Information Theory Applications Workshop, Sand Diego*, 2008.
[13] A. Sridharan and B. Krishnamachari. Maximizing network utilization with max-min fairness in wireless sensor networks. *5th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2007.
[14] A. Sridharan and B. Krishnamachari. Maximizing network utilization with max-min fairness in wireless sensor networks. *Wireless Networks*, 2008.
[15] A.L. Stolyar. Maximizing Queueing Network Utility Subject to Stability: Greedy Primal-Dual Algorithm. *Queueing Systems*, 50(4):401–457, 2005.
[16] L. Tassiulas. Dynamic link activation scheduling in multihop radio networks with fixed or changing topology. 1991.
[17] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and schedulingpolicies for maximum throughput in multihop radio networks. *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 2130–2132, 1990.
[18] 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.
[19] 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.

## APPENDIX A
## LYAPUNOV OPTIMIZATION

Refer to Table I for definitions of variables used in the following work. Let the discrete time queueing equations for

forwarding queues ($U_i(t)$) and virtual queues ($Z_i(t)$) be those defined by update equations (8) and (7) respectively. We define the Lyapunov function as follows:

$$L(\vec{U}(t), \vec{Z}(t)) = \sum U_i^2(t) + \sum Z_i^2(t)$$

Let the Lyapunov drift be represented by $\Delta(\vec{U}(t), \vec{Z}(t))$.

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

The Lyapunov drift can be computed as follows.

Squaring the forwarding queue discrete time queueing equation yields the following:

$$\begin{aligned}U_i^2(t+t) \quad & -U_i^2(t) \leq \\ & -2 \cdot \left[ U_i(t) \cdot \{X_i(t) - \sum_{j \in C_i} X_j(t) - R_i(t)\} \right] \\ & + [X_i(t)]^2 + \left[ \sum_{j \in C_i} X_j(t) + R_i(t) \right]^2 \end{aligned} \tag{13}$$

Note that in typical systems, there exists a bound to the maximum values $X_i(t)$ and $R_i(t)$ can take on. Our problem formulation in Section III limited $X_i(t)$ to at most $B_{max}$. Let the bound on admissions per timeslot be $R_i^{max}$ for node $i$. We'll define constant $G_i$ as follows:

$$\begin{aligned}G_i \quad & \equiv [B_{max}]^2 + \left[ \textstyle\sum_{j \in C_n} B_{max} + R_i^{max} \right]^2 \\ & \geq [X_i(t)]^2 + \left[ \sum_{j \in C_i} X_j(t) + R_i(t) \right]^2 \end{aligned}$$

Similar manipulation can be carried out for the virtual queues.

$$\begin{aligned}Z_i^2(t+t) \quad & -Z_i^2(t) \leq \\ & -2 \cdot \left[ Z_i(t) \cdot \{B_i - \sum_{j \in D_i} X_j(t)\} \right] \\ & + B_i^2 + \left[ \sum_{j \in D_i} X_j(t) \right]^2 \end{aligned} \tag{14}$$

Define constant $K_i$ in a manner similar to $G_i$:

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

Substitution of $G_i$ and $K_i$ into equations (13) and (14), then summing over all nodes $i$, and finally taking the expectation with respect to $(U_i(t), Z_i(t))$, yields the following Lyapunov drift bound:

$$\Delta(\vec{U}(t), \vec{Z}(t)) \leq \Gamma \tag{15}$$

Where

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

Prior work in the field of Lyapunov optimization( [6], [8]) has shown that minimizing Lyapunov drift provides guaranteed stability over system inputs lying within the capacity region. Additionally, any algorithm which minimizes the Lyapunov bound of (15) has been proven to result in system queues that are at worst a constant multiple of the optimal.

As was demonstrated in [6], we can now incorporate a utility function into the drift bound. Let $Y(t) = \sum_i G_i(R_i(t))$ be the system utility, as defined in the problem formulation of Section III. We subtract $V_{opt} \cdot E\{Y(t) | \vec{U}(t), \vec{Z}(t)\}$ from both sides of (15), yielding:

$$\Delta(\cdot) - V_{opt} \cdot E\{Y(t) | \vec{U}(t), \vec{Z}(t)\} \leq \delta \tag{16}$$

Where

$$\begin{aligned}\delta = \quad & \sum_i K_i + \sum_i G_i \\ & -2 \cdot \sum_i \left[ Z_i(t) E\{B_i - \sum_{j \in D_i} X_j(t) \ | Z_i(t)\} \right] \\ & -2 \cdot \sum_i \left[ U_i(t) \cdot E\{X_i(t) - \sum_{j \in C_i} X_j(t) - R_i(t) | U_i(t)\} \right] \\ & -V_{opt} \cdot E\{Y(t) | \vec{U}(t), \vec{Z}(t)\} \end{aligned}$$

In order to minimize the right hand side of (16) in expectation, it is sufficient to ensure we minimize the right hand side for every system state ($\vec{U}(t)$,$\vec{Z}(t)$). We can neglect constant terms involving $K_i$ and $G_i$. The remaining terms can be separated into coefficients multiplying $X_i(t)$ and $R_i(t)$. The goal of our algorithm is then to minimize these terms through intelligent selection of per-timeslot decision variables $X_i(t)$ and $R_i(t)$. As was the case in prior Lyapunov drift work, the resulting algorithm can be broken into two pieces: a transmission control decision and an admission control decision. The control and admission decision are the same as the condition (9) and expression (10) presented in section IV.

*1) Control Decision :* Consider node i with parent node k. The coefficient associated with transmission variable $X_i(t)$ is:

$$-\left[ U_i(t) - \hat{Z}_i(t) - U_k(t) \right] \tag{17}$$

Therefore, if transmission rates $X_i(t)$ and $X_j(t)$ are independent $\forall i, j$, then in order to minimize the right hand side of (16) we maximize $X_i(t)$ $\forall i$ such that (17) is negative. A node therefore transmits data to the parent whenever the differential backlog between the node and its parent exceeds the sum of virtual queues within the local node's neighborhood.

*2) Admission Decision:* Consider node i. The coefficient associated with admission variable $R_i(t)$ is:

$$-\left[\frac{V_{opt}}{2} \cdot g(R_i(t)) - U_i \cdot R_i(t)\right] \tag{18}$$

Therefore, in order to minimize the right hand side of (16) we maximize $R_i(t)$ $\forall i$ such that (18) is negative. This equates to a simple admission control scheme. If the forwarding queue size scaled by admission rate exceeds $\frac{V_{opt}}{2}$ times the utility for all admission rates, then the admission request is rejected. Otherwise, a rate is chosen which maximizes $g(R_i(t))-R_i(t)$.