Handling *Inelastic* Traffic in Wireless Sensor Networks

Jiong Jin*, Avinash Sridharan†, Bhaskar Krishnamachari†, and Marimuthu Palaniswami*
*Dept. of Electrical and Electronic Engineering, The University of Melbourne, Australia
{j.jin, swami}@ee.unimelb.edu.au
†Dept. of Electrical Engineering, University of Southern California, USA
{asridhar, bkrishna}@usc.edu

***Abstract*—The capabilities of sensor networking devices are increasing at a rapid pace. It is therefore not impractical to assume that future sensing operations will involve real time (*inelastic*) traffic, such as audio and video surveillance, which have strict bandwidth constraints. This in turn implies that future sensor networks will have to cater for a mix of *elastic* (having no bandwidth constraint requirements) and inelastic traffic. Current state of the art rate control protocols for wireless sensor networks, are however designed with focus on elastic traffic. In this work, by adapting a recently developed theory of utility-proportional rate control for wired networks to a wireless setting, we present a mathematical framework that gives us elegant queue backpressure-based algorithms. This allows us to design the first-ever rate control protocol that can efficiently handle a mix of elastic and inelastic traffic in a wireless sensor network. The simplicity of our queue backpressure-based algorithm, and the resulting low complexity of its protocol overhead, aids us in demonstrating the implementation of this novel protocol in a real world sensor network stack, the TinyOS-2.x communication stack for IEEE 802.15.4 radios. We evaluate the real-world performance of this protocol through comprehensive experiments on 20 and 40-node subnetworks of USC's 94-node Tutornet wireless sensor network testbed.**

Index Terms — Wireless Sensor Networks, Congestion control, Backpressure-based stacks, Stochastic optimization.

I. Introduction

For low-powers wireless sensor networks (WSNs), the degradation of per-source sustainable rate is quite drastic with increase in network size. In our experiments on the USC Tutornet wireless sensor network testbed [1], it has been observed that with a 40 byte packet, a 4-node network can give per-source rate as low as 16 pkts/sec. A 20-node network under similar conditions results in a reduction of per-source rate to $\sim$ 2 pkts/sec, and in a 40-node network this rate reduces to ~ 0.5 pkts/sec. Thus, even if low data rate sources are operational in these networks (which is typical for a WSN), given the scarce capacity, a sufficiently large number of such sources can easily cause congestion collapse if they are operating unaware of the sustainable rates in the network. This observation reflects the importance of rate control protocols for these networks.

Given the importance of rate control protocols, there have been numerous proposals for congestion control in wireless sensor networks (ARC [2], CODA [3], IFRC [4], RCRT [5], BRCP [6]). All these proposed protocols perform either purely congestion control functionality, or perform congestion control while trying to achieve some form of utility optimization. Despite the variance in the design techniques used, and the capabilities of these protocols, a common theme underlying all these protocols is that they are targeted towards managing traffic, with *elastic* bandwidth requirements. That is, this traffic will present a non-zero utility, as long as it has a non-zero bandwidth allocated to it. Typically, the *goodness* of such a traffic can be measured by defining the utility achieved by the traffic as a logarithmic function of its rate. A typical utility for such traffic is $U_s(r_s) = \log(r_s + 1)$, where U_s is the utility of the source generating the traffic, and r_s is the goodput observed from this source. The utility obtained for an elastic source, at different allocated rates for such a log-based utility can be seen in Figure 1(a).

A key reason for current proposals on rate control protocol design, with focus on elastic traffic, is the nascent nature of sensor network deployments (Habitat monitoring, Structural health monitoring). However, we believe that there are a whole range of applications that are being targeted towards sensor networks, such as audio and video surveillance, real-time traffic monitoring, real-time seismic activity monitoring, that will force sensor networks to support real-time traffic with strict bandwidth constraints. We term such traffic as *inelastic*. The utility for an *inelastic* traffic can be typically given by a sigmoid function. The utility behavior of inelastic traffic with rate is shown in Figure 1(b). Given the heterogeneous nature of sensing devices that nodes in these networks (iMote2f [7], Iris [8], Tmote sky [9]) can carry, we envision that future sensor networks will be required to carry a mix of *elastic* and *inelastic* traffic.

Given that future sensor networks need to have the ability to handle both elastic and inelastic traffic, it is therefore problematic to use state of the art rate control protocols that are designed to support purely elastic traffic. The problem of handling a mix of elastic and inelastic traffic is a known problem in the Internet context. For wired networks, Wang *et al.* [10] have shown that if a mix of elastic and inelastic traffic is run over a network deploying rate control protocols that are designed to optimize the sum utility of elastic traffic, it leads to serious unfairness in terms of and utility performance for the inelastic traffic. Though this observation has been made in the context of wired networks, it is safe to assume that such an un-

Jiong Jin and Avinash Sridharan are equal contributors in this work.

fairness will exist in wireless networks as well. The unfairness, resulting from use of rate control protocols designed for elastic traffic, is primarily due to the underlying design philosophy, which is referred to as "Optimal Flow Control "(OFC). Under OFC, the objective has been to design rate control protocols that will try to maximize the sum utility of the sources. If the utilities are concave Log functions of sources, maximizing sum of utilities will result in a proportional fair rate allocation, making the design principles of OFC particularly attractive when dealing with elastic traffic. However, for inelastic traffic since a proportionally fair rate allocation does not guarantee that the inelastic sources will get a rate greater than their minimum required rate, a proportionally fair rate allocation might result in some inelastic sources getting zero utility; this despite the fact that they have non-zero rates.

For sensor networks, the unfairness resulting from the use of rate control protocols, designed for elastic traffic but used for managing a mix of elastic and inelastic traffic, can have serious detrimental consequences in terms of energy efficiency. Since, though inelastic sources might get non-zero rates with traditional rate-proportional-fair protocols, the packets transmitted to the sink will be useless in adding value to the sensing application. Since they have been transmitted at a rate less than the desired minimum, these packet transmission will thus be wasteful in terms of energy usage, and will drastically impact network life time and application performance.

In the wired context, Wang *et al.* [10], propose a theoretical framework for designing rate control protocols for handling a mix of elastic and inelastic traffic. They show that, if we design rate control protocols that try to achieve proportional fairness in terms of utilities, rather than proportional fairness in terms of rates, it improves the utility performance of inelastic sources drastically, while impacting the utility performance of elastic sources marginally, resulting in overall improvement in system performance.

The following are the key contributions of our work: We show that by merging the theoretical framework presented by Wang *et al.* [10] with a recent proposal on wireless CSMA MAC layer modeling [11], we can develop elegant queue backpressure-based algorithms, that will allow us to design rate control protocols for wireless sensor networks which can optimize of non-concave (inelastic) utilities. To the best of our knowledge this work is a first, in terms of extending the queue backpressure-based stochastic optimization approach for handling fairness of non-concave utility-based traffic in a wireless setting. Furthermore, we do not restrict ourselves to the realm of theory, and use this framework to design and implement a rate control protocol that will be able to support a mix of elastic and inelastic traffic. Our design and implementation of this novel rate control protocol is carried over the TinyOS-2.x communication stack. TinyOS-2.x is one of the most popular operating systems currently used in wireless sensor networks. We test our implementation and design of this new *utility-proportional-fair* rate control protocol over a 20 and 40-node subnetwork of the USC Tutornet testbed [1], a 94 node wireless sensor network, and show that the performance of this novel rate control protocol, over traditional rate control protocols designed for elastic traffic, is much better in terms of the utilities presented to the inelastic traffic, especially when there is a mix of elastic and inelastic traffic in the network.

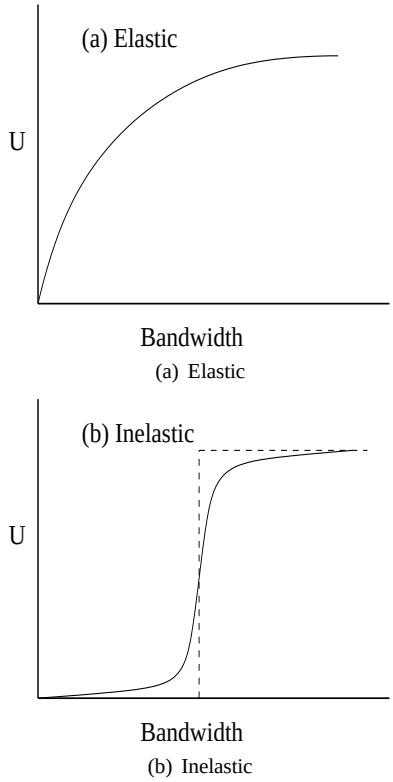


Fig. 1. Behavior of utility for *elastic* and *inelastic* sources.

The rest of the paper is organized as follows. In Section II, we discuss the prior art. In Section III we present our analysis that allows us to extend the framework of wired networks, to a wireless sensor network running over a CSMA MAC. In Section IV, we present the system overview and a software architecture that captures the overall design of the utility-proportional-fair rate control stack over the TinyOS-2.x communication stack. In Section V, we present an evaluation of the utlity-proportional-fair rate control stack presented in Section IV, over the USC Tutornet testbed [1]. Our performance evaluation highlights the gains that this rate control stack presents over traditional designs, when tested in a setting having a mix of elastic and inelastic traffic. Finally, in Section VI, we present our conclusions and future work.

II. Related Work

The formulation of the rate control stack design, as an optimization problem was first motivated by the seminal works of Kelly *et al.* [12] and Low *et al.* [13], in the context of wired networks. Using duality theory, they have proposed distributed algorithms to optimize the network in terms of utility, fairness and stability. Further, Chiang *et al.* [14] have formally presented a mathematical theory of network architecture, and promoted the cross-layer design to optimize system wide network utility. All these results have assumed that the rate-utility function is concave, and thus it is appropriate only for elastic traffic. To address this concern, Wang *et al.* [10] have

developed a new rate control framework of wired networks that is able to deal with both elastic and inelastic traffic, such that the resulting utility is proportional fair.

Alternatively, for wireless networks, Neely *et al.* [15]–[17] have developed a stochastic network optimization framework that models a general network as a queueing system with transmission rate subject to resource allocation decisions, such as scheduling and rate control, to achieve joint optimal performance. The ability to tackle these problems in a stochastic settings makes their solutions more relevant to a wireless multi-hop scenario. In particular, the framework relies heavily on the existence of a backpressure scheduling policy that can be implemented at the MAC layer.

The backpressure scheduling policy was initially proposed by Tassiulas *et al.* [18], and assumes a TDMA synchronized operation with a centralized scheduler. It is proven to achieve the maximum throughput compared with all other scheduling policies. Since the de facto MAC is CSMA in practice, recently there are several attempts to design backpressure-based protocols for CSMA wireless networks. Warrier *et al.* [19] and Akyol *et al.* [20] 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. In addition, Akyol *et al.* [20] also design the flow controllers on top of the scheduler based on the technique proposed by Stolyar [21]. The work by Radunovic *et al.* [22] develops a multi-path routing and rate control protocol, that can be integrated with TCP over 802.11, using backpressure techniques. They use a simple backpressure scheduler that allows transmissions as long as the queue differential is greater than a threshold.

However, none of these proposals are able to divine which backpressure scheduler heuristic should be used to give the best performance in a given setting. On the other hand, Jiang *et al.* [11] shows that under idealized conditions the maximum throughput can be obtained directly on a CSMA MAC by locally prioritizing links with higher queue differentials. Nevertheless, specifically for wireless sensor networks, our latest result [6] reveals and verifies that the gains in implementing queue prioritization over a CSMA MAC are negligible, hence, simple pure backpressure scheduling policy can be implemented for WSNs without modifying the underlying MAC. The novelty of our work is that we show, that by combining the framework presented by Wang *et al.* [10], with the scheduling policy proposed by Jiang *et al.* [11], we can design flow controllers that exhibit proportional fairness in terms of utility. Further, we show that by modifying the scheduling policy proposed by Jiang *et al.* [11], we can achieve the same performance using per-destination queues instead of per-flow queues. This makes the theoretical design of the protocol, practically viable.

There have been several proposals addressing the rate control problem in WSNs [2]–[5], [23]–[25]. Most of these protocols have assumed a clean slate design and follow a router centric, explicit congestion notification approach. A key shortcoming of these protocols, compared to the backpressure-based stacks, is that these protocols are monolithic in nature, optimizing for a specific rate-utility function, or performing pure congestion control without regard to utility optimization. Given the diversity of applications targeted towards sensor networks, it is unclear as to which utility optimization will best serve the requirement of an application. Given the existence of the heterogeneous traffic, both elastic or inelastic rate-utility functions may be observed. In that regards, our backpressure protocol stack, which is able to handle any type of rate-utility function, presents a clear advantage against existing rate control mechanisms in the literature.

III. Analysis

A. An Optimization framework for utility-fair flow control

Rate control (also known as flow control) is an important technique of fair and efficient resource allocation in communication networks. Following the pioneering proposal of optimal flow control [12], an extensive study of network flow control problems has been carried out in the last decade [13], [14], [26], [27]. Essentially, the approach is to formulate flow control as an optimization problem and then maximize the sum utility under the capacity constraint:

$$\textbf{P1:} \quad \max \quad \sum_{s\in\mathcal{S}} U_s(r_s) \tag{1}$$

$$s.t. \qquad r_s \in \Lambda \tag{2}$$

where $\mathcal{S} = \{1, 2, \ldots, S\}$ is the set of sources; for each source $s \in \mathcal{S}$, r_s is the source rate, $U_s(r_s)$ is the associated utility as a measure of performance (or equivalently QoS), and Λ denotes the capacity region.

As shown in [10], even though the optimal flow control (OFC) approach has made great advances in dealing with congestion control and resource allocation, serious limitations still exist due to the objective of OFC of pursuing utility maximization.

- OFC assumes the nature of traffic is elastic (modeled by strictly concave utility functions), therefore it is not applicable for inelastic traffic (modeled by non-concave utility functions) like increasingly popular real-time applications.
- There exists a serious conflict between the utility maximization and utility fairness. For the sources with different QoS requirements, OFC seeks to maximize the total utility, unfortunately this may cause extreme utility unfairness among contending sources. In particular, the source with strict bandwidth demand (*inelastic*) might receive a bandwidth requirement less than its minimum required, resulting in inelastic sources receiving "zero" utility.

In order to support heterogeneous traffic (elastic and inelastic), and guarantee utility fairness among competing flows[1], we will adopt the newly developed flow control strategy by Wang *et al.* [10]. This new mechanism of flow control, which we refer to as *utility-fair* flow control, is not only suitable for elastic traffic but also capable of handling inelastic traffic.

[1]In the remainder of paper, we will use the term flow and source interchangeably.

In this framework, for each source s with utility function $U_s(r_s)$, we define a "pseudo utility" $\mathcal{U}_s(r_s)$ as

$$\mathcal{U}_s(r_s) = \int_{m_s}^{r_s} \frac{1}{U_s(y)} dy, \quad m_s \leq r_s \leq M_s. \tag{3}$$

where $m_s \geq 0$ and $M_s < \infty$ are the minimum and maximum transmission rates required by source s respectively. Now replacing the utility function in the original optimization problem **P1** with "pseudo utility" $\mathcal{U}_s(r_s)$, it leads to the optimization problem **P2**

$$\textbf{P2:} \quad \max \quad \sum_{s\in\mathcal{S}} \mathcal{U}_s(r_s) \tag{4}$$

$$s.t. \qquad r_s \in \Lambda \tag{5}$$

According to the optimization condition of **P2**, at the optimum equilibrium r^*, we have that, for any feasible rate vector $r \neq r^*$,

$$\sum_{s\in\mathcal{S}} \frac{\partial \mathcal{U}_s(r_s^*)}{\partial r_s}(r_s - r_s^*) = \sum_{s\in\mathcal{S}} \frac{r_s - r_s^*}{U_s(r_s^*)} \leq 0 \tag{6}$$

Recalling the definition of well-known *proportional fairness*, a particular bandwidth allocation r^* is *proportional fair* if for any feasible $r \neq r^*$,

$$\sum_{s\in\mathcal{S}} \frac{r_s - r_s^*}{r_s^*} \leq 0. \tag{7}$$

The solution of **P2** thus achieves *utility proportional fairness*.

We would like to emphasize that the strategy (**P2**) is philosophically different from OFC (**P1**); OFC strives to achieve proportional fairness in terms of rates, however, utility-fair flow controller strives to achieve proportional fairness in terms of utilities. Thus, the utility-fair flow controller assumes that the utilities, rather than rates, are a meaningful metrics of QoS for the flows.

Moreover, it is clear from (3) that

$$\mathcal{U}_s'(r_s) = \frac{1}{U(r_s)}, \quad m_s \leq r_s \leq M_s \tag{8}$$

and $\mathcal{U}_s'(r_s) > 0$ is strictly decreasing, according to the strictly increasing property of $U_s(r_s)$. Indeed "pseudo utility" $\mathcal{U}_s(r_s)$ is a strictly increasing concave function in the interval $r_s \in [m_s, M_s]$ regardless the concavity of source s's utility function $U_s(r_s)$. Therefore, **P2** stays as a convex optimization problem after mapping, and is able to deal with elastic and inelastic traffic.

B. Achievable capacity region of CSMA MAC based wireless networks

Since the optimization framework of utility-fair flow control is primarily developed for wired networks, it cannot be directly applied to wireless networks. The fundamental difference is that the capacity region Λ for wireless networks depends very much on the underlying MAC layer. In order to solve the problem $\mathbf{P2}$ in a wireless setting, we need a model that presents us well all the constraints for the problem $\mathbf{P2}$ in a wireless setting. One way of achieving this is to formulate a scheduling policy that will achieve the optimal throughput rate region of the wireless network. The scheduling policy will then give us the constraints for the problem $\mathbf{P2}$.

Recently, Jiang *et al.* [11] proposes an adaptive scheduling algorithm that can theoretically achieve the maximum throughput over a CSMA MAC. The work shows that if the rate of the back-off window (an exponential random variable) is chosen according to a queue differential between a node and its parent, the system can achieve 100% throughput. We can therefore use the framework proposed by Jiang *et al.* [11], to define the constraints of the problem $\mathbf{P2}$. In the next section, we will combine the above frameworks proposed by Wang *et al.* [10], and Jiang *et al.* [11], to develop a queue backpressure-based utility-fair flow control protocol that will be implemented at the transport and the MAC layer respectively. We formulate the problem by taking a cross-layer approach, and propose a joint flow control and CSMA scheduling algorithm. Notice that the formulation and analysis is based on general multi-hop wireless networks, where sensor networks is a special case.

A key problem with the scheduling policy proposed by Jiang *et al.* [11] is that it assumes the per-flow queue model, i.e., each link maintains a separate queue for each flow traversing through that link. This will obviously result in overhead, and make the implementation difficult, especially when a large number of flows are active in the network. Therefore, when merging the frameworks proposed by Wang *et al.* [10] and Jiang *et al.* [11], we show that the scheduling policy proposed by Jiang *et al.* [11] can be changed from *per-flow queue* to *per-destination queue*, without any loss in performance. This modification particularly helps in simplifying the implementation of the protocol.

C. Joint flow control and scheduling algorithm

Consider a multi-hop wireless network with L links, and let $\mathcal{L}$ denote the set of links and $\mathcal{N}$ denote the set of nodes. For each link $l \in \mathcal{L}$, it can also be represented by a node pair (i, j), meaning transmission from node i to node j. Assume there are a set $\mathcal{S}$ of flows and a set $\mathcal{D}$ of intended destinations. Each flow s is associated with a source node f_s and a destination node d_s, and is able to adjust the input rate. Let r_s be the rate, and $\mathcal{U}_s(r_s)$ be the defined "pseudo utility" function of flow s.

Each node i maintains per-destination queues for flows that traverse it, i.e., different queues are maintained for flows destined to different destinations. Denote γ_{ij}^d the service rate of queue towards $d \in \mathcal{D}$ on link (i, j). Then, the service rate of each particular queue on node i should be no less than the incoming rate, i.e., $\sum_{j:(i,j)\in\mathcal{L}} \gamma_{ij}^d \geq \sum_{j:(j,i)\in\mathcal{L}} \gamma_{ji}^d + \sum_{s:f_s=i,d_s=d} r_s$.

Furthermore, in [11], it shows that the maximum-weight scheduling policy can be effectively modeled as maximizing the entropy of a Markov chain

$$\begin{aligned} \max_{u} \quad & -\sum_k u_k \log u_k \\ s.t. \quad & \sum_k u_k \cdot x_{(i,j)}^k \geq \lambda_{ij}, \quad \forall (i,j) \in \mathcal{L} \end{aligned} \tag{9}$$

where u_k is a variable satisfying $u_k \geq 0, \sum_k u_k = 1$. $x_{(i,j)}^k$ indicates the transmission status of link (i, j) under a particular

scheduler $x^k \in \{0,1\}^L$. λ_{ij} is normalized i.i.d arrival traffic rate at each link (i,j).

Now, we can formulate the following optimization problem:

$$\begin{aligned}
\max_{u,\gamma,r} \quad & -\sum_k u_k \log u_k + V \sum_{s\in\mathcal{S}} \mathcal{U}_s(r_s) \qquad (10)\\
s.t. \quad & \gamma_{ij}^d \geq 0, \quad \forall (i,j) \in \mathcal{L}, \forall d \in \mathcal{D}\\
& \sum_{j:(i,j)\in\mathcal{L}} \gamma_{ij}^d \geq \sum_{j:(j,i)\in\mathcal{L}} \gamma_{ji}^d + \sum_{\substack{s:f_s=i\\ d_s=d}} r_s, \ \forall d \in \mathcal{D}, \forall i \neq d\\
& \sum_k u_k \cdot x_{(i,j)}^k = \sum_{d\in\mathcal{D}} \gamma_{ij}^d, \quad \forall (i,j) \in \mathcal{L}\\
& u_k \geq 0, \sum_k u_k = 1.
\end{aligned}$$

The tuning parameter V is used in (10) to determine the extent to which "pseudo utility" optimization is emphasized. As shown in [17], it presents a tradeoff between the optimality and system queue backlog. The 3rd constraint of (10) says that the total service rate of link (i,j) is divided among the destination queues. By associating dual variables $q_i^d \geq 0$ to the 2nd constraint, a partial Lagrangian (subject to $\gamma_{ij}^d \geq 0, \sum_k u_k \cdot x_{(i,j)}^k = \sum_{d\in\mathcal{D}} \gamma_{ij}^d$ and $u_k \geq 0, \sum_k u_k = 1$) is

$$\begin{aligned}
& \mathcal{L}(u,\gamma,r;q)\\
= \quad & -\sum_k u_k \log(u_k) + V \sum_{s\in\mathcal{S}} \mathcal{U}_s(r_s)\\
& + \sum_{d\in\mathcal{D}, i\neq d} q_i^d \left(\sum_{j:(i,j)\in\mathcal{L}} \gamma_{ij}^d - \sum_{j:(j,i)\in\mathcal{L}} \gamma_{ji}^d - \sum_{\substack{s:f_s=i\\ d_s=d}} r_s \right)\\
= \quad & -\sum_k u_k \log(u_k) + V \sum_{s\in\mathcal{S}} \mathcal{U}_s(r_s)\\
& - \sum_{s\in\mathcal{S}} q_{f_s}^{d_s} r_s + \sum_{(i,j)\in\mathcal{L}, d\in\mathcal{D}} \gamma_{ij}^d (q_i^d - q_j^d) \qquad (11)
\end{aligned}$$

First fixing the vectors u and q, we solve for γ_{ij}^d in the sub-problem

$$\begin{aligned}
\max_{\gamma} \quad & \sum_{(i,j)\in\mathcal{L}, d\in\mathcal{D}} \gamma_{ij}^d (q_i^d - q_j^d)\\
s.t. \quad & \gamma_{ij}^d \geq 0, \quad \forall (i,j) \in \mathcal{L}, \forall d \in \mathcal{D}\\
& \sum_{d\in\mathcal{D}} \gamma_{ij}^d = \sum_k u_k \cdot x_{(i,j)}^k, \quad \forall (i,j) \in \mathcal{L}. \qquad (12)
\end{aligned}$$

It is quite straightforward to find the solution: for each link (i,j), let $d^*(i,j) = \arg\max_d (q_i^d - q_j^d)$, and let $\gamma_{ij}^d = \sum_k u_k \cdot x_{(i,j)}^k$ if $d = d^*(i,j)$ and $\gamma_{ij}^d = 0$, otherwise. In other words, each link schedules the transmission of the destination queue whose backpressure $q_i^d - q_j^d$ is maximum.

Substitute the solution of (12) into (11), we get

$$\begin{aligned}
\mathcal{L}(u,r;q) = & \left[-\sum_k u_k \log(u_k) + \sum_{(i,j)\in\mathcal{L}} z_{ij} \left(\sum_k u_k \cdot x_{(i,j)}^k \right) \right]\\
& + \left[V \sum_{s\in\mathcal{S}} \mathcal{U}_s(r_s) - \sum_{s\in\mathcal{S}} q_{f_s}^{d_s} r_s \right]
\end{aligned}$$

where $z_{ij} := \max_d (q_i^d - q_j^d)$ is the maximum backpressure of link (i,j). Hence, a distributive algorithm to solve (10) is as follows

Algorithm: Joint flow control and scheduling

Initially, assume that all queues are empty, and set $q_i^d = 0, \forall i, d$.

- **Transport layer (flow control)**: the rate of each flow s is determined by

$$\begin{aligned}
r_s^* = \quad & argmax_{r_s} \{V \cdot \mathcal{U}_s(r_s) - q_{f_s}^{d_s} r_s\}\\
= \quad & U_s^{-1} \left(\frac{V}{q_{f_s}^{d_s}} \right).
\end{aligned}$$

  It maximizes $\mathcal{L}(u,r;q)$ over r.
- **MAC layer (scheduling)**: link (i,j) schedules the transmission of destination queue with the maximum backpressure $z_{ij} = \max_d (q_i^d - q_j^d)$ when it gets the opportunity. Specifically, the back-off window size is set to be an exponential random variable with mean $\frac{1}{\exp(z_{ij})}$. It maximizes $\mathcal{L}(u,r;q)$ over u [11]. The dual variable q_i^d is updated by

$$q_i^d \leftarrow \left[q_i^d - \alpha \left(\sum_{j:(i,j)\in\mathcal{L}} \gamma_{ij}^d - \sum_{j:(j,i)\in\mathcal{L}} \gamma_{ji}^d - \sum_{\substack{s:f_s=i\\ d_s=d}} r_s \right) \right]^+$$

  where $[a]^+ = \max\{0, a\}$. We observe that $q_i^d \propto Q_i^d$. Then Q_i^d, the actual queue length of node i for destination d, can be used as the corresponding dual variable.

Remark 1: Though the Algorithm appears similar to those in [17], [21], it is philosophically different in terms of the objective and the chosen MAC model.

IV. Implementing utility-fair flow control in TinyOS-2.x

As mentioned earlier, the objective of this work is not only to build a framework that will allow us to present quantitative designs of utility-fair flow controllers that can support inelastic traffic in a sensor network, but to realize these designs in practice. In this section, inline with our goals, we present a software architecture that allows us to implement the utility-fair controller presented in Section III, in the TinyOS-2.x network stack. We choose TinyOS-2.x as our target platform, since it is one of the most popular operating systems used in sensor networks.

As described in Section III-C, in order to implement the utility-fair flow controller we need to implement a backpressure-based scheduler at the MAC layer, and use the queueing information presented by this backpressure-based scheduler to implement the utility-fair flow controllers at the transport layer. Figure 2 presents a software architecture that captures the design of such a backpressure-based rate control stack

For the purposes of this work 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. We concentrate specifically on a collection

tree, since as shown in Section III, it is trivial to extend this design to multiple destination. In order to support multiple destination all that needs to be added to this design is a per-destination queue.

When routing is fixed, the backpressure-based rate control stack is implemented at the MAC and the transport layers. The transport layer functionality is implemented as part of the "Leaky Bucket" and "Flow Controller" blocks in Figure 2. 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 2 interacts with the forwarding engine to learn the instantaneous queue size, and sets an allowed admission rate in the leaky bucket. The leaky bucket then generates tokens at the admission rate. When a packet arrives from the application to the flow controller, it is injected into the forwarding engine only if a token is available.

The backpressure-based MAC is implemented as part of the "Forwarding Engine" and "Communication stack" blocks (Figure 2). 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 2). If the scheduler wants to implement differential queue prioritization, the forwarding engine can use interfaces provided by the underlying MAC to modify the MAC back-off window sizes, before injecting the packet.

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

A. Transport layer

The key component in implementing the transport layer is the flow controller block. The objective of the flow controller is to operate the source at a time average rate r_s, allowing the source to achieve a utility $U_s(r_s)$, such that the rate allocation $r_s,\ \forall\ s$, maximizes $\sum_s U_s(r_s)$ across the entire network. Note that the flow controller runs at each node, and hence it needs to make local decisions, but the local decisions should be such as to optimize a global function $(\max\ \sum_{\forall s} U_s(r_s))$.

If we want to implement a utility-fair flow controller, Section III shows that it maximizes the total "pseudo utility" $\mathcal{U}_s(r_s) = \int \frac{1}{U_s(r_s)} dr_s$. For an inelastic source, the utility function $U_s(r_s)$ will be given by a *sigmoid* function, while for the elastic source the utility function $U_s(r_s)$ will be given by a *logarithmic* function [28]. We now present the design of the sigmoid-utility-fair flow controller and the log-utility-fair flow controller, for regulating inelastic and elastic traffic sources.

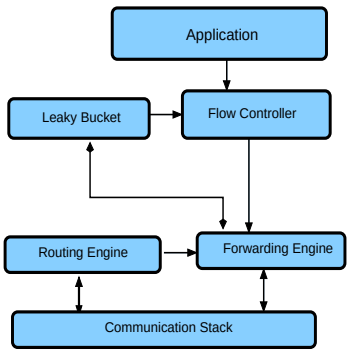


Fig. 2. Software architecture for a backpressure-based stack.

1) Sigmoid-utility-fair flow controller: The sigmoid-utility-fair flow controller is designed to be used with an inelastic source. The utility function for real-time inelastic traffic is as follows:

$$\begin{aligned} U_s(r_s) &= 0,\ \text{if } r_s \leq B_{min} \\ &= \tfrac{1}{1+e^{-a(r_s-b)}}, \text{if } B_{min} \leq r_s \leq B_{max} \\ &= 1,\ \text{if } r_s \geq B_{max} \end{aligned}$$

B_{min} and B_{max} are the minimum and maximum bandwidth constraints on the sigmoid, $b = (\frac{B_{max}-B_{min}}{2}) + B_{min}$, a controls the slope of the sigmoid.

From the Algorithm in Section III-C, the optimal rate r_s^* is given by:

$$r_s^* = argmax\left(V\mathcal{U}_s(r_s) - q_s r_s\right) = U_s^{-1}\left(\frac{V}{q_s}\right)$$

Note that since here we consider the single destination case, the superscript d is omitted for simplicity. As pointed out, V is a constant that acts as a tuning parameter to effect a tradeoff between the forwarding queue size q_s, and "pseudo utility" $\mathcal{U}_s(r_s)$. A large value of V will imply large value of q_s, and large total $\mathcal{U}_s(r_s)$. Whereas a small value of V will imply small value of q_s, and small total $\mathcal{U}_s(r_s)$.[2] The implementation would set r_s^* as follows:

$$\begin{aligned} r_s^* &= b - \tfrac{1}{a}\log\left(\tfrac{q_s}{V} - 1\right),\ q_s > V \\ &= B_{max},\ q_s < V \end{aligned}$$

2) Log-utility-fair flow controller: The log-utility-fair flow controller is designed to be used with elastic traffic. The utility function for elastic traffic is as follows:

$$U_s(r_s) = \log(r_s + 1)$$

We offset the rate by $+1$ to ensure the positiveness of the utility. Again, the optimal rate r_s^* is

$$r_s^* = argmax\left(V\mathcal{U}_s(r_s) - q_s r_s\right) = U_s^{-1}\left(\frac{V}{q_s}\right)$$

Hence,

$$r_s^* = e^{\frac{V}{q_s}} - 1$$

B. MAC Layer

Ideally, we should be implementing the scheduling policy by Algorithm in Section III-C, that can theoretically achieve the maximum throughput on a CSMA MAC. This requires implementing a backoff window at the MAC layer, where the value of the backoff window is chosen as an exponential random variable with a mean inverse proportional to the queue differential between the transmitting node and its parent. While theoretically the scheme initially proposed by Jiang *et al.* [11]

[2]It should be noted that the flow controller designs presented here are similar to the proposals presented by Sridharan *et al.* [6].

is quite appealing, in practice it is hard to realize every possible mapping between the queue differential of a node with its parent, and the resulting backoff window mechanism. This is primarily due to the fact, that in theory the backoff window size is a real number and can potentially go to zero; in practice, however, due to hardware and software limitations we have to limit the window size to a practically viable minimum and maximum limits. In [6] the authors show that due to these practical limitations, in theory, a CSMA MAC whose window size is mapped to the queue differential can perform much worse than a much simpler backpressure scheme, where the forwarding engine allows a packet to the enter the MAC layer if it sees a positive queue differential with the parent. The authors, in [6], call this naive version of a backpressure scheduling policy, the positive differential queue MAC (PDQ MAC). Based on the empirical evidence presented by us in [6], we therefore choose to implement the backpressure scheduling policy at the MAC using the PDQ MAC.

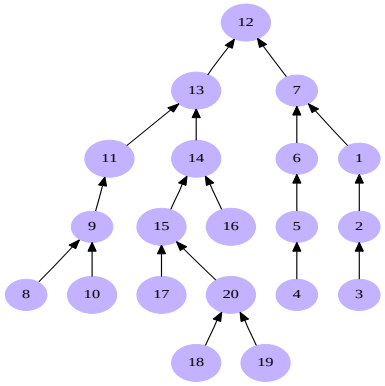
V. Evaluation

In order to test the performance of the *utility-fair flow controllers* presented in Section IV-A and implemented in TinyOS-2.x, we ran different types of traffic generators (elastic and inelastic) over the backpressure-based rate control stack (Section IV). We compared its performance with a similar backpressure-based stack running the *proportional-fair flow controller* instead of the utility-fair flow controllers. The proportional-fair flow controller is the traditional flow controller, based on the OFC design, where the flow controller tries to optimize for the sum utility $(\sum_s U_s(r_s))$ of the traffic instead of trying to optimize for the sum "pseudo utility" $\left(\sum_s \mathcal{U}_s(r_s) = \sum_s \int \frac{1}{U_i(r_s)} dr_s\right)$, which is the objective of the utility-fair flow controller. The purpose of this comparative evaluation is two fold: first it shows how using an OFC based flow controller, such as the proportional-fair flow controller leads to extremely poor performance for *inelastic* traffic, when inelastic and elastic traffic are mixed; second it shows the gains achieved, in terms of improvements in utility performance of *inelastic* traffic, when used in the same traffic settings.

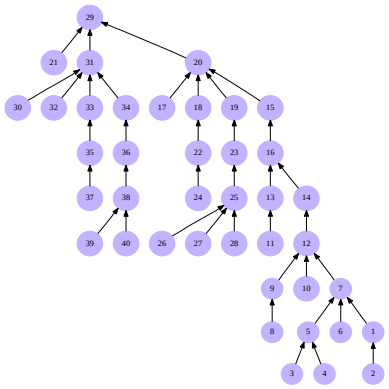
A. Experimental setup

The comparative experiments were performed on the USC Tutornet testbed [1], a 94 node wireless sensor network testbed. The testbed consists of Tmote sky [9] devices which can run the TinyOS-2.x operating system. The Tmote sky devices come fitted with an 802.15.4 compatible CC2420 [29] radio. The TinyOS-2.x platform comes with a default CSMA MAC, called the CC2420 MAC, that can operate over these radios. Of the 94 nodes present in the testbed, we used a maximum of 40 nodes. The experiments were performed over two different routing topologies, a 20-node topology and a 40-node topology shown in Figure 3(a) and 3(b).

The CC2420 radio can operate at 31 different power levels. For these experiments, for the 20-node topology, we operated the CC2420 radios at a power level of 5, and at a power level of 10 for the 40-node topologies. The connectivity graph, for the 20-node and 40-node topologies, are given in Figure 4. A

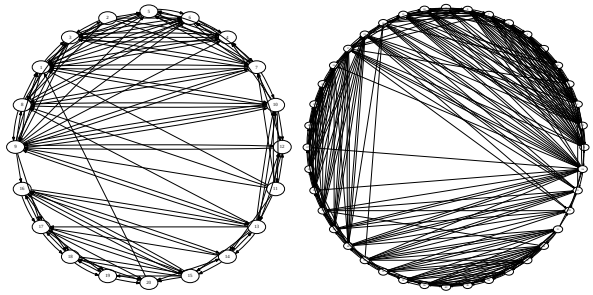


(a) 20-node



(b) 40-node

Fig. 3. Routing topologies on the USC Tutornet [1] testbed.



(a) 20-node (Power = 5) (b) 40-node (Power = 10)

Fig. 4. Connectivity for 20 and 40-node topologies on the USC tutornet testbed. Links indicate PRR of at least 80%.

link between two nodes is shown in the connectivity graph if they have packet success probability greater than 80%. The connectivity graphs gives us an idea of the density of deployment, and also justifies the choice of power levels for these topologies, since it shows that the graphs are connected at the chosen power levels.

B. Proportional-fair flow controller

The proportional-fair flow controller is designed and implemented similar to the utility-fair flow controllers, which have been described in Section IV-A. The only difference between the implementation of the utility-fair flow controllers and the proportional-fair flow controller is that in the case of the utility-fair flow controller we deal with "pseudo utility"

$\mathcal{U}_s(r_s) = \int \frac{1}{U_s(r_s)} dr_s$, where as for the proportional flow controller we just deal with the utility $U_s(r_s)$.

The flow controller than sets the instantaneous rate $r_s(t)$ to the following :

$$r_s^* = argmax\,(VU_s(r_s) - q_s r_s)$$

For the proportional flow controller $U_s(r_s) = \log(r_s + 1)$, hence the instantaneous rate will be set to :

$$\begin{aligned} r_s^* &= \tfrac{V}{q_s} - 1, \text{ if } \tfrac{V}{q_s} \geq 1 \\ &= V, \text{ if } q_s = 0 \\ &= 0, \text{ if } \tfrac{V}{q_s} < 1 \end{aligned}$$

where V is the same constant that presents a tradeoff between the total utility achieved and the queue size in the system. Incidentally, this is the exact design that has been proposed by Sridharan *et al.* in [6] and [30], using the Lyapunov drift-based stochastic optimization technique proposed by Neely *et al.* [15]–[17], as well as the design proposed by Akyol *et al* [20] based on the stochastic optimization technique proposed by Stolyar [21].

C. Traffic sources

A key to performing this empirical evaluation is to have traffic generators that can emulate the elastic and inelastic traffic in a real world wireless sensor network. For *elastic* traffic generator, we choose a CBR traffic that generates a packet every 10 ms. Emulating an inelastic source is more complicated. To emulate the inelastic source, we implemented a traffic generator that injects packets into the system as follows: recall that the utility of an inelastic source is given by a sigmoid function having a minimum and maximum bandwidth constraints of B_{min} and B_{max}. Our inelastic source simply tries to emulate this utility function; if the allocated rate to the source is less than B_{min} the source does not inject any packets into the system; if the allocated rate to the source is $B_{min} \leq r_i < B_{max}$, the source injects packet into the system with a probability $p = \frac{1}{1+e^{-a(r_s-b)}}$, where $b = (\frac{B_{max}-B_{min}}{2}) + B_{min}$, and a is set to 2; if the allocated rate to the source is greater than B_{max} it injects packet into the system probability 1. The inelastic source generates packets at a constant rate of 1 packet every 10 ms, however whether it decides to inject the packet into the system or not depends on the above mentioned conditions. The packet sizes in our network are ~ 40 bytes.

D. Parameter selection

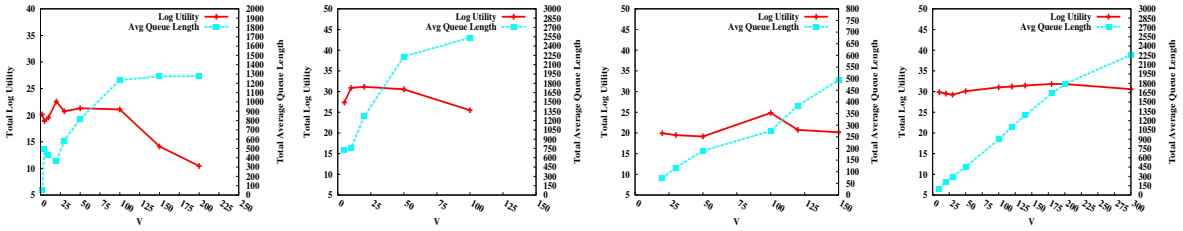
As remarked in section III, the performance of the backpressure-based stack depends on the parameter V. The parameter V presents a tradeoff between the queue size in the system and how much the optimality achieved. A large V will force the system to operate close to edge of the rate region; allowing for close-to-optimal utility, albeit at the cost of large queues. While a small value of V will allow the system to operate well within the rate region; allowing for small queue sizes, albeit at the cost of optimality in terms of utility.

For the 20 and 40-node topologies, in order to determine the optimal setting of V that should be used with each of the flow controllers, we plot the utility and queue behavior for different values of V. For this experiment all sources are assumed to be *elastic* sources. Thus the system utility being measured is $\sum_s \log(r_s)$. Since all sources are *elastic*, we use the *log-utility* fair flow controller (Section IV-A2) when testing the utility-fair flow controllers. The results of this experiment are presented in Figure 5. For each of the flow controllers, over each of the topologies, it can be seen that the utility increases logarithmically with V, while queue sizes increases linearly with V. Also, for each of the graphs, after a certain value of V the utility actually starts falling while the queue sizes keep increasing. This behavior occurs due the finite queue sizes that exist in all practical system. Due to the limitation of queue sizes, packet drops start occurring after a certain value of V resulting in loss of utility. Thus, for the system to operate efficiently we will have to select a V that will allow for good utility while allowing the system to operate within the finite bounds of the queue sizes. From this figure it can be seen that a value of $V = 30$ for the 20-node topology, and $V = 5$ for the 40-node topology will present good performance for the utility-fair flow controller. Similarly for the proportional flow controller we choose a value of $V = 100$ for the 20-node topology, and $V = 50$ for the 40-node topology

E. *Performance of the Utility-fair flow controller*

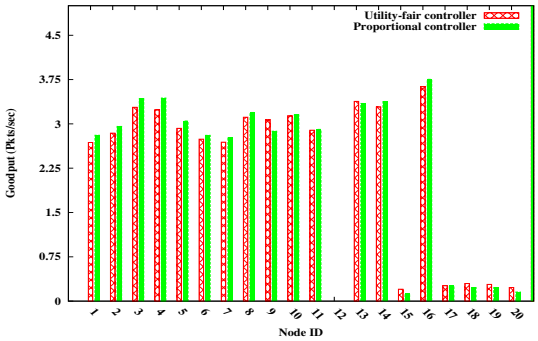
We perform a comparative evaluation between the two types of flow controllers using the following two traffic scenarios: In the first scenario all sources in the network are *elastic*. This particular scenario helps us validate the implementation of the utility-fair controller. In this scenario, the utility and goodput for the sources should be similar to the utility and goodput, when using the proportional flow controller. For the second scenario, we use traffic mix of *elastic* and *inelastic* traffic. This particular scenario showcases the advantage of the utility-fair flow controller over the proportional fair flow controller. The expectation in this scenario is that the utility-fair flow controller will treat the inelastic traffic fairly, by giving it a higher priority, and presenting the inelastic traffic with a better utility than in the case when the same traffic mix is run over a proportional flow controller.

*1) **Elastic Traffic**:* The Figure 6 and Figure 7 present the performance of the utility-fair flow controller and the proportional-fair flow controller, when all sources in the network are elastic. Since all sources are elastic, all nodes use the *log-utility* flow controller to represent the utility-fair flow controller. For both, the 20-node and 40-node topologies, it can be seen that the goodput distribution using either flow controller is very similar. The proportional flow controller outperforms the log-utility flow controller, but by only a small margin. For the 40-node topology the proportional flow controller presents a sum log-utility of 31.22, while the log-utility flow controller presents a sum log-utility of 27.36. The gap between the performance of the two flow controllers reduces even further for the 20-node topology. For the 20-node topology the sum log-utility presented by the proportional flow controller is 20.74, while that presented by the log-utility flow controller is 20.73. The reason for the sub-optimality

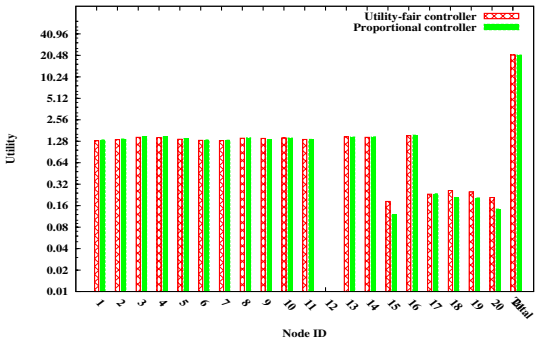


(a) 20 node, Utility-fair controller (b) 40 node, Utility-fair controller (c) 20 node, proportional controller (d) 40 node, proportional controller

Fig. 5. Selecting the optimal value of the parameter V for the 20 and 40-node topologies, for the utility-fair and proportional fair flow controllers.

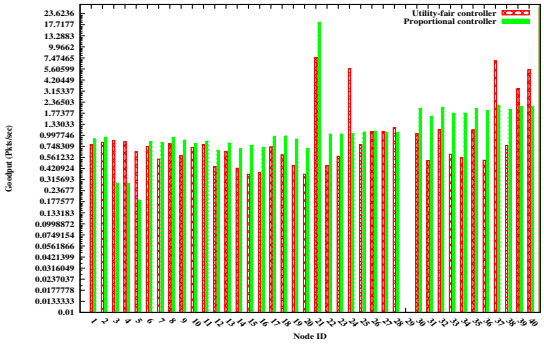


(a) Goodput

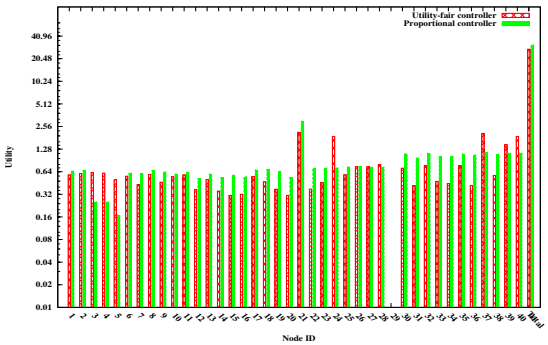


(b) Utility

Fig. 6. Goodput and utility comparison for the 20-node topology with elastic traffic.



(a) Goodput

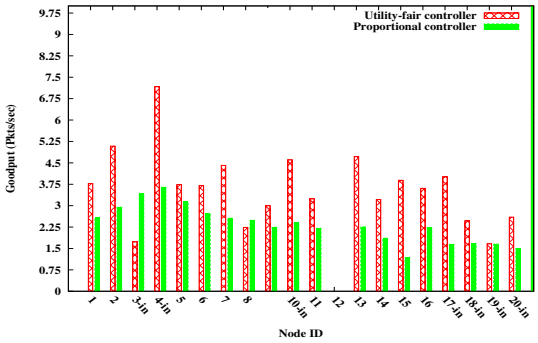


(b) Utility

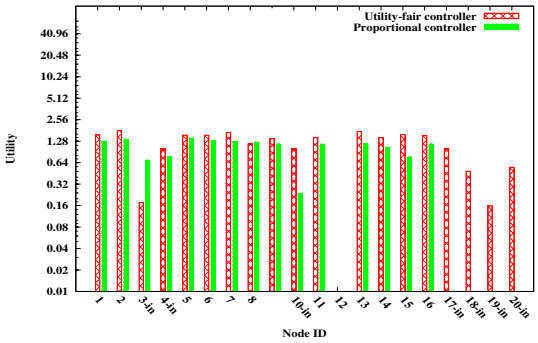
Fig. 7. Goodput and utility comparison for the 40-node topology with elastic traffic.

of the log-utility-fair controller is that, the log-utility-fair controller has been optimized for presenting a proportionally fair solution in terms of utility; while the proportional fair controller is designed specifically to maximize the sum log-utility, since this presents proportional fairness in terms of rates. The objective of presenting these results is that though the log-utility-fairness is suboptimal in terms of achieving proportional fairness in terms of rates, it still is able to give a rate distribution comparative to the proportional-fair controller. Since, as will be shown in the next section, the utility-fair flow controller gives a distinct advantage over proportional flow controller when a mix of elastic and inelastic traffic exist; the results presented in this section motivate the argument for the use of the utility-fair flow controller in all traffic scenarios, given that in pure elastic traffic settings the cost in terms of performance, when using the utility-fair controller, is marginal.

*2) **Elastic and Inelastic Traffic**:* The goal of developing the utility-fair flow controllers was to present nodes in a wireless sensor network the ability to support a mix of *elastic* and *inelastic* traffic. In this section we validate this goal. We test the performance of the utility-fair flow controller and the proportional-fair flow controller over the 20 and 40-node topologies, in terms of the goodputs and utility achieved by the sources, under a mixed traffic setting. In order to emulate the mix of elastic and inelastic traffic, for the 20-node topology sources 3, 4, 10, 17, 18, 19 and 20 are inelastic sources; all other sources are elastic sources. For the 40-node topology sources 1-10, 12, 26, 27, 28, 39 and 40 are inelastic; all other sources are elastic sources. As mentioned earlier, in the case of the utility-fair flow controllers the source use a specific type of utility-fair flow controller, depending on the type of traffic they are generating. The elastic source use the *log-utility* flow controller, and the inelastic sources use the *sigmoid-utility* flow

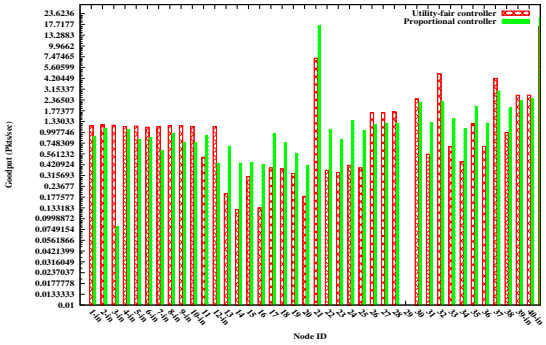


(a) Goodput

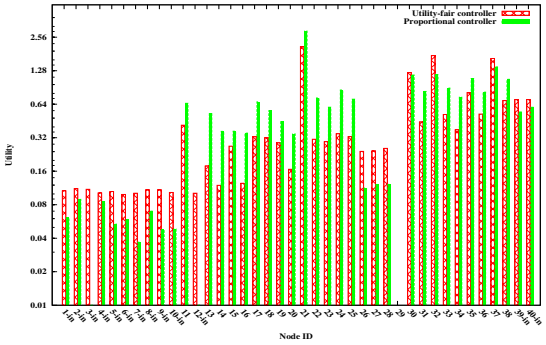


(b) Utility

Fig. 8. Goodput and utility comparison for the utility-fair flow controller, and the proportional flow controller for the 20-node topology. For inelastic traffic $B_{min} = 2$ pkts/sec, and $B_{max} = 4$ pkts/sec.



(a) Goodput



(b) Utility

Fig. 9. Goodput and utility comparison for the utility-fair flow controller, and the proportional flow controller for the 40-node topology. For inelastic traffic $B_{min} = 1$ pkts/sec, and $B_{max} = 4$ pkts/sec.

controller. For the case of the proportional flow controller, since there exist only a single type of flow controller, all sources use the same flow controller.

There is a specific reason for choosing this distribution of sources for these topologies. Since the proportional flow controller strives to achieve a tradeoff between efficiency and fairness, it presents sources that are closer to the sink with a much higher rate than nodes that are farther away from the sink. Thus, by choosing the given traffic mix we are able to clearly highlight the disdvantage the proportional flow controller presents to the inelastic traffic. This is reflected in the results for the 20 and 40-node topologies shown in Figures 8 and 9.

For the 20-node topology, for the inelastic traffic the B_{min} of the sigmoid utility is set to 2 pkts/sec, and B_{max} is set to 4 pkts/sec. For the 40-node topology, for the inelastic traffic the B_{min} of the sigmoid utility is set to 1 pkts/sec, and B_{max} is set to 4 pkts/sec. In the x-axis, the inelastic traffic sources are marked with the suffix "-in". Both the goodput and utility plots show that, when using the proportional fair flow controller some of the inelastic sources get a goodput less than B_{min}. A direct result of this unfairness results in those inelastic sources getting zero utility. In a practical sense this implies that any data sent by these sources simply resulted in a wastage of energy since the data received at the sink will be useless, given that they are getting zero utility. The performance of the inelastic sources is greatly improved when using the utility-fair controllers. As can be seen when using the utility-fair flow controllers, the inelastic sources get a much higher goodput, since they are given higher priority, and this automatically results in a much higher utility than in the case of the proportional fair flow controller. The results presented in this section clearly validate our design and motivation of using utility-fair controllers for handling a mix of elastic and inelastic traffic.

VI. Conclusion and Future Work

In this work, we have extended the concept of designing rate control protocols that can achieve utility-proportional fairness in a wireless sensor network running over a CSMA MAC. Experiments of this novel rate control stack, over the USC Tutornet [1] testbed, show that this new rate control stack presents inelastic sources with much better utilities, than if the same sources were run with a rate-proportional fair rate control protocol, designed using the traditional optimal flow control model.

Though the empirical results presented here are encouraging, there are still some open problem before the protocols presented can be widely adopted in existing communication stacks. The key issue is with the parameter V, on which the performance of these protocols rely heavily. As mentioned earlier, and seen in our empirical results, V presents a tradeoff between the network queue size and utilities achieved. The optimal setting of V depends on the traffic pattern (number of flows in the network), and the network topology. Thus, depending on the traffic pattern and network topology, the

setting of V might need to be changed continuously in order to derive good performance from the stack. This has been highlighted in [6] as well. Our future work will therefore be targeted towards designing online algorithms that can estimate on the fly, the value of this parameter V.

REFERENCES

[1] "http://testbed.usc.edu."

[2] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *Proceedings of the 7th annual International Conference on Mobile Computing and Networking*, 2001, pp. 221–235.

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

[4] 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.

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

[6] A. Sridharan, S. Moeller, and B. Krishnamachari, "Implementing backpressure-based rate control protocols for wireless sensor networks," submitted to ACM Mobicom 2009.

[7] "http://www.xbow.com/products/productdetails.aspx?sid=253."

[8] "http://www.xbow.com/products/productdetails.aspx?sid=264."

[9] "http://www.moteiv.com."

[10] W. H. Wang, M. Palaniswami, and S. H. Low, "Application-oriented flow control: fundamentals, algorithms and fairness," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1282–1291, December 2006.

[11] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," in *Allerton Conference on Communication, Control, and Computing*, 2008.

[12] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[13] S. H. Low and D. E. Lapsley, "Optimization flow control—I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, December 1999.

[14] 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.

[15] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[16] ——, "Energy optimal control for time varying wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915–2934, July 2006.

[17] M. J. Neely, E. Modiano, and C. P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 396–409, April 2008.

[18] 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*, vol. 37, no. 12, pp. 1936–1949, December 1992.

[19] A. Warrier, L. Le, and I. Rhee, "Cross-layer optimization made practical," *Broadnets*, invited Paper.

[20] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Sanjee, and A. Stolyar, "Joint scheduling and congestion control in mobile ad-hoc networks," in *Proceedings of IEEE INFOCOM 2008*, April 2008, pp. 1292–1300.

[21] A. L. Stolyar, "Maximizing queueing network utility subject to stability: greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, 2005.

[22] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: Balancing tcp over multiple paths in wireless mesh networks," in *Proceedings of ACM Mobicom 2008*, September 2008.

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

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

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

[26] R. La and V. Anantharam, "Utility based rate control in the Internet for elastic traffic," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 272–286, April 2002.

[27] X. Lin and N. B. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 766–781, May 2006.

[28] S. Shenker, "Fundamental design issues for the future Internet," *jsac*, vol. 13, no. 7, pp. 1176–1188, July 1995.

[29] *CC2420, True single-chip 2.4 GHz IEEE 802.15.4/ZigBee RF transceiver with MAC support*, Chipcon Inc, http://www.chipcon.com.

[30] A. Sridharan, S. Moeller, and B. Krishnamachari, "Making distributed rate control using Lyapunov drift a reality in wireless sensor networks," in *Proceedings of WiOpt*, 2008.